Lecture Notes in Computer Science 3515

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
New York University, NY, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Vaidy S. Sunderam Geert Dick van Albada
Peter M.A. Sloot Jack J. Dongarra (Eds.)

Computational Science – ICCS 2005

5th International Conference
Atlanta, GA, USA, May 22-25, 2005
Proceedings, Part II

Springer

Volume Editors

Vaidy S. Sunderam
Emory University
Dept. of Math and Computer Science
400 Dowman Dr, W430, Atlanta, GA 30322, USA
E-mail: vss@mathcs.emory.edu

Geert Dick van Albada
Peter M.A. Sloot
University of Amsterdam
Department of Mathematics and Computer Science
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
E-mail: {dick,sloot}@science.uva.nl

Jack J. Dongarra
University of Tennessee
Computer Science Departement
1122 Volunteer Blvd., Knoxville, TN 37996-3450, USA
E-mail: dongarra@cs.utk.edu

Library of Congress Control Number: 2005925759

CR Subject Classification (1998): D, F, G, H, I, J, C.2-3

ISSN 0302-9743
ISBN-10 3-540-26043-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26043-1 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11428848 06/3142 5 4 3 2 1 0

Preface

The Fifth International Conference on Computational Science (ICCS 2005) held in Atlanta, Georgia, USA, May 2225, 2005, continued in the tradition of previous conferences in the series: ICCS 2004 in Krakow, Poland; ICCS 2003 held simultaneously at two locations, in Melbourne, Australia and St. Petersburg, Russia; ICCS 2002 in Amsterdam, The Netherlands; and ICCS 2001 in San Francisco, California, USA.

Computational science is rapidly maturing as a mainstream discipline. It is central to an ever-expanding variety of fields in which computational methods and tools enable new discoveries with greater accuracy and speed. ICCS 2005 was organized as a forum for scientists from the core disciplines of computational science and numerous application areas to discuss and exchange ideas, results, and future directions. ICCS participants included researchers from many application domains, including those interested in advanced computational methods for physics, chemistry, life sciences, engineering, economics and finance, arts and humanities, as well as computer system vendors and software developers. The primary objectives of this conference were to discuss problems and solutions in all areas, to identify new issues, to shape future directions of research, and to help users apply various advanced computational techniques. The event highlighted recent developments in algorithms, computational kernels, next generation computing systems, tools, advanced numerical methods, data-driven systems, and emerging application fields, such as complex systems, finance, bioinformatics, computational aspects of wireless and mobile networks, graphics, and hybrid computation. Keynote lectures were delivered by John Drake – High End Simulation of the Climate and Development of Earth System Models; Marian Bubak – Recent Developments in Computational Science and the CrossGrid Project; Alok Choudhary – Scientific Data Management; and David Keyes – Scientific Discovery through Advanced Computing.

In addition, four invited presentations were delivered by representatives of industry: David Barkai from Intel Corporation, Mladen Karcic from IBM, Tom Rittenberry from SGI and Dan Fay from Microsoft. Seven tutorials preceded the main technical program of the conference: Tools for Program Analysis in Computational Science by Dieter Kranzlmüller and Andreas Knüpfer; Computer Graphics and Geometric Modeling by Andrés Iglesias; Component Software for High Performance Computing Using the CCA by David Bernholdt; Computational Domains for Explorations in Nanoscience and Technology, by Jun Ni, Deepak Srivastava, Shaoping Xiao and M. Meyyappan; Wireless and Mobile Communications by Tae-Jin Lee and Hyunseung Choo; Biomedical Literature Mining and Its Applications in Bioinformatics by Tony Hu; and Alternative Approaches to Grids and Metacomputing by Gunther Stuer; We would like to thank all keynote, invited and tutorial speakers for their interesting and inspiring talks.

Aside from the plenary lectures, the conference included 10 parallel oral sessions and 3 poster sessions. Ever since the first meeting in San Francisco, ICCS has attracted an increasing number of researchers involved in the challenging field of computational science. For ICCS 2005, we received 464 contributions for the main track and over 370 contributions for 24 originally-proposed workshops. Of these submissions, 134 were accepted as full papers accompanied by oral presentations, and 89 for posters in the main track, while 241 papers were accepted for presentations at 21 workshops. This selection was possible thanks to the hard work of the 88-member Program Committee and 362 reviewers. The author index contains 1395 names, and over 500 participants from 41 countries and all continents attended the conference. The ICCS 2005 proceedings consists of three volumes. The first volume, LNCS 3514 contains the full papers from the main track of the conference, while volumes 3515 and 3516 contain the papers accepted for the workshops and short papers. The papers cover a wide range of topics in computational science, ranging from numerical methods, algorithms, and computational kernels to programming environments, grids, networking and tools. These contributions, which address foundational and computer science aspects are complemented by papers discussing computational applications in a variety of domains. ICCS continues its tradition of printed proceedings, augmented by CD-ROM versions. We would like to thank Springer-Verlag for their cooperation and partnership. We hope that the ICCS 2005 proceedings will serve as a major intellectual resource for computational science researchers for many years to come. During the conference the best papers from the main track and workshops as well as the best posters were nominated and commended on the ICCS 2005 Website. A number of papers will also be published in special issues of selected journals.

We owe thanks to all workshop organizers and members of the Program Committee for their diligent work, which led to the very high quality of the event. We would like to express our gratitude to Emory University and Emory College in general, and the Department of Mathematics and Computer Science in particular, for their wholehearted support of ICCS 2005. We are indebted to all the members of the Local Organizing Committee for their enthusiastic work towards the success of ICCS 2005, and to numerous colleagues from various Emory University units for their help in different aspects of organization. We very much appreciate the help of Emory University students during the conference. We owe special thanks to our corporate sponsors: Intel, IBM, Microsoft Research, SGI, and Springer-Verlag; and to ICIS, Math & Computer Science, Emory College, the Provost's Office, and the Graduate School at Emory University for their generous support. ICCS 2005 was organized by the Distributed Computing Laboratory at the Department of Mathematics and Computer Science at Emory University, with support from the Innovative Computing Laboratory at the University of Tennessee and the Computational Science Section at the University of Amsterdam, in cooperation with the Society for Industrial and Applied Mathematics (SIAM). We invite you to visit the ICCS 2005 Website (http://www.iccs-meeting.org/ICCS2005/) to recount the events leading up to the conference, to

view the technical program, and to recall memories of three and a half days of engagement in the interest of fostering and advancing Computational Science.

June 2005 Vaidy Sunderam, Scientific Chair, ICCS 2005

on behalf of the co-editors:
G. Dick van Albada, Workshops Chair, ICCS 2005
Jack J. Dongarra, ICCS Series Overall co-Chair
Peter M.A. Sloot, ICCS Series Overall Chair

Organization

ICCS 2005 was organized by the Distributed Computing Laboratory, Department of Mathematics and Computer Science, Emory University, Atlanta, GA, USA. in cooperation with Emory College, Emory University (USA), the University of Tennessee (USA), the University of Amsterdam (The Netherlands), and the Society for Industrial and Applied Mathematics (SIAM). The conference took place on the campus of Emory University, in Atlanta, Georgia, USA.

Conference Chairs

Scientific Chair - Vaidy Sunderam (Emory University, USA)
Workshops Chair - Dick van Albada (University of Amsterdam, The Netherlands)
ICCS Series Overall Chair - Peter M.A. Sloot (University of Amsterdam, The Netherlands)
ICCS Series Overall Co-Chair - Jack Dongarra (University of Tennessee, USA)

Local Organizing Committee

Dawid Kurzyniec (Chair)
Piotr Wendykier
Jeri Sandlin
Erin Nagle
Ann Dasher
Sherry Ebrahimi

Sponsoring Institutions

Intel Corporation IBM Corporation
Microsoft Research SGI Silicon Graphics Inc.
Emory University, Department of Mathematics and Computer Science
Emory University, Institute for Comparative and International Studies
Emory University, Emory College
Emory University, Office of the Provost
Emory University, Graduate School of Arts and Sciences
Springer-Verlag

Program Committee

Jemal Abawajy, Deakin University, Australia
David Abramson, Monash University, Australia
Dick van Albada, University of Amsterdam, The Netherlands
Vassil Alexandrov, University of Reading, UK
Srinivas Aluru, Iowa State University, USA
Brian d'Auriol, University of Texas at El Paso, USA
David A. Bader, University of New Mexico, USA
Saeid Belkasim, Georgia State University, USA
Anne Benoit, University of Edinburgh, UK
Michele Benzi, Emory University, USA
Rod Blais, University of Calgary, Canada
Alexander Bogdanov, Institute for High Performance Computing and Information Systems, Russia
Anu Bourgeois, Georgia State University, USA
Jan Broeckhove, University of Antwerp, Belgium
Marian Bubak, Institute of Computer Science and ACC Cyfronet - AGH, Poland
Rajkumar Buyya, University of Melbourne, Australia
Tiziana Calamoneri, University of Rome "La Sapienza", Italy
Serge Chaumette, University of Bordeaux, France
Toni Cortes, Universitat Politecnica de Catalunya, Spain
Yiannis Cotronis, University of Athens, Greece
Jose C. Cunha, New University of Lisbon, Portugal
Pawel Czarnul, Gdansk University of Technology, Poland
Frederic Desprez, INRIA, France
Tom Dhaene, University of Antwerp, Belgium
Hassan Diab, American University of Beirut, Lebanon
Beniamino Di Martino, Second University of Naples, Italy
Jack Dongarra, University of Tennessee, USA
Craig Douglas, University of Kentucky, USA
Edgar Gabriel, University of Stuttgart, Germany
Marina Gavrilova, University of Calgary, Canada
Michael Gerndt, Technical University of Munich, Germany
Yuriy Gorbachev, Institute for High Performance Computing and Information Systems, Russia
Andrzej Goscinski, Deakin University, Australia
Eldad Haber, Emory University, USA
Ladislav Hluchy, Slovak Academy of Science, Slovakia
Alfons Hoekstra, University of Amsterdam, The Netherlands
Yunqing Huang, Xiangtan University, China
Andrés Iglesias, University of Cantabria, Spain
Hai Jin, Huazhong University of Science and Technology, China
Peter Kacsuk, MTA SZTAKI Research Institute, Hungary
Jacek Kitowski, AGH University of Science and Technology, Poland

Dieter Kranzlmüller, Johannes Kepler University Linz, Austria
Valeria Krzhizhanovskaya, University of Amsterdam, The Netherlands
Dawid Kurzyniec, Emory University, USA
Domenico Laforenza, Italian National Research Council, Italy
Antonio Lagana, Universita di Perugia, Italy
Francis Lau, The University of Hong Kong, P.R. China
Laurent Lefevre, INRIA, France
Bogdan Lesyng, ICM Warszawa, Poland
Thomas Ludwig, University of Heidelberg, Germany
Emilio Luque, University Autonoma of Barcelona, Spain
Piyush Maheshwari, University of New South Wales, Australia
Maciej Malawski, Institute of Computer Science AGH, Poland
Michael Mascagni, Florida State University, USA
Taneli Mielikäinen, University of Helsinki, Finland
Edward Moreno, Euripides Foundation of Marilia, Brazil
Wolfgang Nagel, Dresden University of Technology, Germany
Genri Norman, Russian Academy of Sciences, Russia
Stephan Olariu, Old Dominion University, USA
Salvatore Orlando, University of Venice, Italy
Robert M. Panoff, Shodor Education Foundation, Inc, USA
Marcin Paprzycki, Oklahoma State University, USA
Ron Perrott, Queen's University of Belfast, UK
Richard Ramaroson, ONERA, France
Rosemary Renaut, Arizona State University, USA
Alistair Rendell, Australian National University, Australia
Paul Roe, Queensland University of Technology, Australia
Dale Shires, U.S. Army Research Laboratory, USA
Charles Shoniregun, University of East London, UK
Magda Slawinska, Gdansk University of Technology, Poland
Peter Sloot, University of Amsterdam, The Netherlands
Gunther Stuer, University of Antwerp, Belgium
Boleslaw Szymanski, Rensselaer Polytechnic Institute, USA
Ryszard Tadeusiewicz, AGH University of Science and Technology, Poland
Pavel Tvrdik, Czech Technical University, Czech Republic
Putchong Uthayopas, Kasetsart University, Thailand
Jesus Vigo-Aguiar, University of Salamanca, Spain
Jerzy Wasniewski, Technical University of Denmark, Denmark
Greg Watson, Los Alamos National Laboratory, USA
Peter H. Welch, University of Kent , UK
Piotr Wendykier, Emory University, USA
Roland Wismüller, University of Siegen, Germany
Baowen Xu, Southeast University Nanjing, China
Yong Xue, Chinese Academy of Sciences, China
Xiaodong Zhang, College of William and Mary, USA
Alexander Zhmakin, SoftImpact Ltd, Russia

Krzysztof Zielinski, ICS UST / CYFRONET, Poland
Zahari Zlatev, National Environmental Research Institute, Denmark
Elena Zudilova-Seinstra, University of Amsterdam, The Netherlands

Reviewers

Adrian Kacso
Adrian Sandu
Akshaye Dhawan
Alberto Sanchez-Campos
Alex Tiskin
Alexander Bogdanov
Alexander Zhmakin
Alexandre Dupuis
Alexandre Tiskin
Alexandros Gerbessiotis
Alexey S. Rodionov
Alfons Hoekstra
Alfredo Tirado-Ramos
Ali Haleeb
Alistair Rendell
Ana Ripoll
A. Kalyanaraman
Andre Merzky
Andreas Hoffmann
Andrés Iglesias
Andrew Adamatzky
Andrzej Czygrinow
Andrzej Gościński
Aneta Karaivanova
Anna Morajko
Anne Benoit
Antonio Lagana
Anu G. Bourgeois
Ari Rantanen
Armelle Merlin
Arndt Bode
B. Frankovic
Bahman Javadi
Baowen Xu
Barbara Głut
Bartosz Baliś
Bas van Vlijmen
Bastien Chopard
Behrooz Shirazi
Ben Jackson
Beniamino Di Martino
Benjamin N. Jackson
Benny Cheung
Biju Sayed
Bogdan Lesyng
Bogdan Smolka
Boleslaw Szymanski
Breanndan O'Nuallain
Brian d'Auriol
Brice Goglin
Bruce Boghosian
Casiano Rodrguez León
Charles Shoniregun
Charles Stewart
Chen Lihua
Chris Homescu
Chris R. Kleijn
Christian Glasner
Christian Perez
C. Schaubschlaeger
Christoph Anthes
Clemens Grelck
Colin Enticott
Corrado Zoccolo
Craig C. Douglas
Craig Lee
Cristina Negoita
Dacian Daescu
Daewon W. Byun
Dale Shires
Danica Janglova
Daniel Pressel
Dave Roberts
David Abramson
David A. Bader
David Green
David Lowenthal
David Roberts
Dawid Kurzyniec
Dick van Albada
Diego Javier Mostaccio
Dieter Kranzlmüller
Dirk Deschrijver
Dirk Roekaerts
Domenico Laforenza
Donny Kurniawan
Eddy Caron
Edgar Gabriel
Edith Spiegl
Edward Moreno
Eldad Haber
Elena Zudilova-Seinstra
Elisa Heymann
Emanouil Atanassov
Emilio Luque
Eunjoo Lee
Eunjung Cho
Evarestov
Evghenii Gaburov
Fabrizio Silvestri
Feng Tan
Fethi A. Rabhi
Floros Evangelos
Francesco Moscato
Francis Lau
Francisco J. Rosales
Franck Cappello
Frank Dehne
Frank Dopatka
Frank J. Seinstra
Frantisek Capkovic
Frederic Desprez
Frederic Hancke

Frédéric Gava
Frédéric Loulergue
Frederick T. Sheldon
Gang Kou
Genri Norman
George Athanasopoulos
Greg Watson
Gunther Stuer
Haewon Nam
Hai Jin
Hassan Diab
He Jing
Holger Bischof
Holly Dail
Hongbin Guo
Hongquan Zhu
Hong-Seok Lee
Hui Liu
Hyoung-Key Choi
Hyung-Min Lee
Hyunseung Choo
I.M. Navon
Igor Mokris
Igor Schagaev
Irina Schweigert
Irina Shoshmina
Isabelle Guérin-Lassous
Ivan Dimov
Ivana Budinska
J. Kroc
J.G. Verwer
Jacek Kitowski
Jack Dongarra
Jan Broeckhove
Jan Glasa
Jan Humble
Jean-Luc Falcone
Jean-Yves L'Excellent
Jemal Abawajy
Jens Gustedt
Jens Volkert
Jerzy Waśniewski
Jesus Vigo-Aguiar
Jianping Li
Jing He
Jinling Yang
John Copeland
John Michopoulos
Jonas Latt
Jongpil Jeong
Jose L. Bosque
Jose C. Cunha
Jose Alberto Fernandez
Josep Jorba Esteve
Jun Wu
Jürgen Jähnert
Katarzyna Rycerz
Kawther Rekabi
Ken Nguyen
Ken C.K. Tsang
K.N. Plataniotis
Krzysztof Boryczko
Krzysztof Grzda
Krzysztof Zieliński
Kurt Vanmechelen
Ladislav Hluchy
Laurence T. Yang
Laurent Lefevre
Laurent Philippe
Lean Yu
Leigh Little
Liang Cheng
Lihua Chen
Lijuan Zhu
Luis M. Portela
Luoding Zhu
M. Mat Deris
Maciej Malawski
Magda Sławińska
Marcin Paprzycki
Marcin Radecki
Marcin Smtek
Marco Aldinucci
Marek Gajcki
Maria S. Pérez
Marian Bubak
Marina Gavrilova
Marios Dikaiakos
Martin Polak
Martin Quinson
Massiomo Coppola
Mathilde Romberg
Mathura Gopalan
Matthew Sottile
Matthias Kawski
Matthias Müller
Mauro Iacono
Michał Malafiejski
Michael Gerndt
Michael Mascagni
Michael Navon
Michael Scarpa
Michele Benzi
Mikhail Zatevakhin
Miroslav Dobrucky
Mohammed Yousoof
Moonseong Kim
Moshe Sipper
Nageswara S. V. Rao
Narayana Jayaram
NianYan
Nicola Tonellotto
Nicolas Wicker
Nikolai Simonov
Nisar Hundewale
Osni Marques
Pang Ko
Paul Albuquerque
Paul Evangelista
Paul Gray
Paul Heinzlreiter
Paul Roe
Paula Fritzsche
Paulo Afonso Lopes
Pavel Tvrdik
Paweł Czarnul
Paweł Kaczmarek
Peggy Lindner
Peter Brezany
Peter Hellinckx
Peter Kacsuk
Peter Sloot
Peter H. Welch
Philip Chan
Phillip A. Laplante

Pierre Fraigniaud
Pilar Herrero
Piotr Bala
Piotr Wendykier
Piyush Maheshwari
Porfidio Hernandez
Praveen Madiraju
Putchong Uthayopas
Qiang-Sheng Hua
R. Vollmar
Rafał Wcisło
Rafik Ouared
Rainer Keller
Rajkumar Buyya
Rastislav Lukac
Renata Słota
Rene Kobler
Richard Mason
Richard Ramaroson
Rob H. Bisseling
Robert M. Panoff
Robert Schaefer
Robin Wolff
Rocco Aversa
Rod Blais
Roeland Merks
Roland Wismüller
Rolf Rabenseifner
Rolf Sander
Ron Perrott
Rosemary Renaut
Ryszard Tadeusiewicz
S. Lakshmivarahan
Saeid Belkasim
Salvatore Orlando
Salvatore Venticinque
Sam G. Lambrakos
Samira El Yacoubi
Sang-Hun Cho
Sarah M. Orley
Satoyuki Kawano
Savio Tse
Scott Emrich
Scott Lathrop
Seong-Moo Yoo
Serge Chaumette
Sergei Gorlatch
Seungchan Kim
Shahaan Ayyub
Shanyu Tang
Sibel Adali
Siegfried Benkner
Sridhar Radharkrishnan
Srinivas Aluru
Srinivas Vadrevu
Stefan Marconi
Stefania Bandini
Stefano Marrone
Stephan Olariu
Stephen Gilmore
Steve Chiu
Sudip K. Seal
Sung Y. Shin
Takashi Matsuhisa
Taneli Mielikäinen
Thilo Kielmann
Thomas Ludwig
Thomas Richter
Thomas Worsch
Tianfeng Chai
Timothy Jones
Tiziana Calamoneri
Todor Gurov
Tom Dhaene
Tomasz Gubała
Tomasz Szepieniec
Toni Cortes
Ulrich Brandt-Pollmann
V. Vshivkov
Vaidy Sunderam
Valentina Casola
V. Krzhizhanovskaya
Vassil Alexandrov
Victor Malyshkin
Viet D. Tran
Vladimir K. Popkov
V.V. Shakhov
Włodzimierz Funika
Wai-Kwong Wing
Wei Yin
Wenyuan Liao
Witold Alda
Witold Dzwinel
Wojtek Gościński
Wolfgang E. Nagel
Wouter Hendrickx
Xiaodong Zhang
Yannis Cotronis
Yi Peng
Yong Fang
Yong Shi
Yong Xue
Yumi Choi
Yunqing Huang
Yuriy Gorbachev
Zahari Zlatev
Zaid Zabanoot
Zhenjiang Hu
Zhiming Zhao
Zoltan Juhasz
Zsolt Nemeth

Workshops Organizers

High Performance Computing in Academia: Systems and Applications

Denis Donnelly - Siena College, USA
Ulrich Rüde - Universität Erlangen-Nürnberg

Tools for Program Development and Analysis in Computational Science

Dieter Kranzlmüller - GUP, Joh. Kepler University Linz, Austria
Arndt Bode - Technical University Munich, Germany
Jens Volkert - GUP, Joh. Kepler University Linz, Austria
Roland Wismüller - University of Siegen, Germany

Practical Aspects of High-Level Parallel Programming (PAPP)

Frédéric Loulergue - Université Paris Val de Marne, France

2005 International Workshop on Bioinformatics Research and Applications

Yi Pan - Georgia State University, USA
Alex Zelikovsky - Georgia State University, USA

Computer Graphics and Geometric Modeling, CGGM 2005

Andrés Iglesias - University of Cantabria, Spain

Computer Algebra Systems and Applications, CASA 2005

Andrés Iglesias - University of Cantabria, Spain
Akemi Galvez - University of Cantabria, Spain

Wireless and Mobile Systems

Hyunseung Choo - Sungkyunkwan University, Korea
Eui-Nam Huh Seoul - Womens University, Korea
Hyoung-Kee Choi - Sungkyunkwan University, Korea
Youngsong Mun - Soongsil University, Korea

Intelligent Agents in Computing Systems -The Agent Days 2005 in Atlanta

Krzysztof Cetnarowicz - Academy of Science and Technology AGH, Krakow, Poland
Robert Schaefer - Jagiellonian University, Krakow, Poland

Programming Grids and Metacomputing Systems - PGaMS2005

Maciej Malawski - Institute of Computer Science, Academy of Science and Technology AGH, Krakow, Poland
Gunther Stuer - Universiteit Antwerpen, Belgium

Autonomic Distributed Data and Storage Systems Management - ADSM2005

Jemal H. Abawajy - Deakin University, Australia
M.Mat Deris - College University Tun Hussein Onn, Malaysia

GeoComputation

Yong Xue - London Metropolitan University, UK

Computational Economics and Finance

Yong Shi - University of Nebraska, Omaha, USA
Xiaotie Deng - University of Nebraska, Omaha, USA
Shouyang Wang - University of Nebraska, Omaha, USA

Simulation of Multiphysics Multiscale Systems

Valeria Krzhizhanovskaya - University of Amsterdam, The Netherlands
Bastien Chopard - University of Geneva, Switzerland
Yuriy Gorbachev - Institute for High Performance Computing & Data Bases, Russia

Dynamic Data Driven Application Systems

Frederica Darema - National Science Foundation, USA

2nd International Workshop on Active and Programmable Grids Architectures and Components (APGAC2005)

Alex Galis - University College London, UK

Parallel Monte Carlo Algorithms for Diverse Applications in a Distributed Setting

Vassil Alexandrov - University of Reading, UK
Aneta Karaivanova - Institute for Parallel Processing, Bulgarian Academy of Sciences
Ivan Dimov - Institute for Parallel Processing, Bulgarian Academy of Sciences

Grid Computing Security and Resource Management

Maria Pérez - Universidad Politécnica de Madrid, Spain
Jemal Abawajy - Deakin University, Australia

Modelling of Complex Systems by Cellular Automata

Jiri Kroc - Helsinki School of Economics, Finland
S. El Yacoubi - University of Perpignan, France
M. Sipper - Ben-Gurion University, Israel
R. Vollmar - University Karlsruhe, Germany

International Workshop on Computational Nano-Science and Technology

Jun Ni - The University of Iowa, USA
Shaoping Xiao - The University of Iowa, USA

New Computational Tools for Advancing Atmospheric and Oceanic Sciences

Adrian Sandu - Virginia Tech, USA

Collaborative and Cooperative Environments

Vassil Alexandrov - University of Reading, UK
Christoph Anthes - GUP, Joh. Kepler University Linz, Austria
David Roberts - University of Salford, UK
Dieter Kranzlmüller - GUP, Joh. Kepler University Linz, Austria
Jens Volkert - GUP, Joh. Kepler University Linz, Austria

Table of Contents

Workshop On "High Performance Computing in Academia: Systems and Applications"

Teaching High-Performance Computing on a High-Performance Cluster
Martin Bernreuther, Markus Brenk, Hans-Joachim Bungartz, Ralf-Peter Mundani, Ioan Lucian Muntean 1

Teaching High Performance Computing Parallelizing a Real Computational Science Application
Giovanni Aloisio, Massimo Cafaro, Italo Epicoco, Gianvito Quarta . . 10

Introducing Design Patterns, Graphical User Interfaces and Threads Within the Context of a High Performance Computing Application
James Roper, Alistair P. Rendell 18

High Performance Computing Education for Students in Computational Engineering
Uwe Fabricius, Christoph Freundl, Harald Köstler, Ulrich Rüde 27

Integrating Teaching and Research in HPC: Experiences and Opportunities
M. Berzins, R.M. Kirby, C.R. Johnson 36

Education and Research Challenges in Parallel Computing
L. Ridgway Scott, Terry Clark, Babak Bagheri 44

Academic Challenges in Large-Scale Multiphysics Simulations
Michael T. Heath, Xiangmin Jiao 52

Balancing Computational Science and Computer Science Research on a Terascale Computing Facility
Calvin J. Ribbens, Srinidhi Varadarjan, Malar Chinnusamy, Gautam Swaminathan 60

Computational Options for Bioinformatics Research in Evolutionary Biology
Michael A. Thomas, Mitch D. Day, Luobin Yang 68

Financial Computations on Clusters Using Web Services
Shirish Chinchalkar, Thomas F. Coleman, Peter Mansfield 76

"Plug-and-Play" Cluster Computing HPC Designed for the Mainstream Scientist
Dean E. Dauger, Viktor K. Decyk 84

Building an HPC Watering Hole for Boulder Area Computational Science
E.R. Jessup, H.M. Tufo, M.S. Woitaszek 91

The Dartmouth Green Grid
James E. Dobson, Jeffrey B. Woodward, Susan A. Schwarz, John C. Marchesini, Hany Farid, Sean W. Smith 99

Resource-Aware Parallel Adaptive Computation for Clusters
James D. Teresco, Laura Effinger-Dean, Arjun Sharma 107

Workshop on "Tools for Program Development and Analysis in Computational Science"

New Algorithms for Performance Trace Analysis Based on Compressed Complete Call Graphs
Andreas Knüpfer and Wolfgang E. Nagel 116

PARADIS: Analysis of Transaction-Based Applications in Distributed Environments
Christian Glasner, Edith Spiegl, Jens Volkert 124

Automatic Tuning of Data Distribution Using Factoring in Master/Worker Applications
Anna Morajko, Paola Caymes, Tomàs Margalef, Emilio Luque 132

DynTG: A Tool for Interactive, Dynamic Instrumentation
Martin Schulz, John May, John Gyllenhaal 140

Rapid Development of Application-Specific Network Performance Tests
Scott Pakin 149

Providing Interoperability for Java-Oriented Monitoring Tools with JINEXT
Włodzimierz Funika, Arkadiusz Janik 158

RDVIS: A Tool That Visualizes the Causes of Low Locality and Hints Program Optimizations
Kristof Beyls, Erik H. D'Hollander, Frederik Vandeputte 166

CacheIn: A Toolset for Comprehensive Cache Inspection
Jie Tao, Wolfgang Karl 174

Optimization-Oriented Visualization of Cache Access Behavior
Jie Tao, Wolfgang Karl 182

Collecting and Exploiting Cache-Reuse Metrics
Josef Weidendorfer, Carsten Trinitis 191

Workshop on "Computer Graphics and Geometric Modeling, CGGM 2005"

Modelling and Animating Hand Wrinkles
X.S. Yang, Jian J. Zhang 199

Simulating Wrinkles in Facial Expressions on an Anatomy-Based Face
Yu Zhang, Terence Sim, Chew Lim Tan 207

A Multiresolutional Approach for Facial Motion Retargetting Using Subdivision Wavelets
Kyungha Min, Moon-Ryul Jung 216

New 3D Graphics Rendering Engine Architecture for Direct Tessellation of Spline Surfaces
Adrian Sfarti, Brian A. Barsky, Todd J. Kosloff, Egon Pasztor, Alex Kozlowski, Eric Roman, Alex Perelman 224

Fast Water Animation Using the Wave Equation with Damping
Y. Nishidate, G.P. Nikishkov 232

A Comparative Study of Acceleration Techniques for Geometric Visualization
Pascual Castelló, José Francisco Ramos, Miguel Chover 240

Building Chinese Ancient Architectures in Seconds
Hua Liu, Qing Wang, Wei Hua, Dong Zhou, Hujun Bao 248

Accelerated 2D Image Processing on GPUs
Bryson R. Payne, Saeid O. Belkasim, G. Scott Owen, Michael C. Weeks, Ying Zhu 256

Consistent Spherical Parameterization
Arul Asirvatham, Emil Praun, Hugues Hoppe 265

Mesh Smoothing via Adaptive Bilateral Filtering
Qibin Hou, Li Bai, Yangsheng Wang 273

Towards a Bayesian Approach to Robust Finding Correspondences in Multiple View Geometry Environments
Cristian Canton-Ferrer, Josep R. Casas, MontsePardàs 281

Managing Deformable Objects in Cluster Rendering
Thomas Convard, Patrick Bourdot, Jean-Marc Vézien 290

Revolute Quadric Decomposition of Canal Surfaces and Its Applications
Jinyuan Jia, Ajay Joneja, Kai Tang 298

Adaptive Surface Modeling Using a Quadtree of Quadratic Finite Elements
G. P. Nikishkov 306

MC Slicing for Volume Rendering Applications
A. Benassarou, E. Bittar, N. W. John, L. Lucas 314

Modelling and Sampling Ramified Objects with Substructure-Based Method
Weiwei Yin, Marc Jaeger, Jun Teng, Bao-Gang Hu 322

Integration of Multiple Segmentation Based Environment Models
SeungTaek Ryoo, CheungWoon Jho 327

On the Impulse Method for Cloth Animation
Juntao Ye, Robert E. Webber, Irene Gargantini 331

Remeshing Triangle Meshes with Boundaries
Yong Wu, Yuanjun He, Hongming Cai 335

SACARI: An Immersive Remote Driving Interface for Autonomous Vehicles
Antoine Tarault, Patrick Bourdot, Jean-Marc Vézien 339

A 3D Model Retrieval Method Using 2D Freehand Sketches
Jiantao Pu, Karthik Ramani 343

A 3D User Interface for Visualizing Neuron Location in Invertebrate Ganglia
Jason A. Pamplin, Ying Zhu, Paul S. Katz, Rajshekhar Sunderraman 347

Workshop on "Modelling of Complex System by Cellular Automata"

The Dynamics of General Fuzzy Cellular Automata
Angelo B. Mingarelli 351

A Cellular Automaton SIS Epidemiological Model with Spatially Clustered Recoveries
David Hiebeler 360

Simulating Market Dynamics with CD++
Qi Liu, Gabriel Wainer 368

A Model of Virus Spreading Using Cell-DEVS
Hui Shang, Gabriel Wainer 373

A Cellular Automata Model of Competition in Technology Markets with Network Externalities
Judy Frels, Debra Heisler, James Reggia, Hans-Joachim Schuetze ... 378

Self-organizing Dynamics for Optimization
Stefan Boettcher 386

Constructibility of Signal-Crossing Solutions in von Neumann 29-State Cellular Automata
William R. Buckley, Amar Mukherjee 395

Evolutionary Discovery of Arbitrary Self-replicating Structures
Zhijian Pan, James Reggia 404

Modelling Ant Brood Tending Behavior with Cellular Automata
Daniel Merkle, Martin Middendorf, Alexander Scheidler 412

A Realistic Cellular Automata Model to Simulate Traffic Flow at Urban Roundabouts
Ruili Wang, Mingzhe Liu 420

Probing the Eddies of Dancing Emergence: Complexity and Abstract Painting
Tara Krause 428

Workshop on "Wireless and Mobile Systems"

Enhanced TCP with End-to-End Bandwidth and Loss Differentiation Estimate over Heterogeneous Networks
Le Tuan Anh, Choong Seon Hong 436

Content-Aware Automatic QoS Provisioning for UPnP AV-Based Multimedia Services over Wireless LANs
Yeali S. Sun, Chang-Ching Yan, Meng Chang Chen 444

Simulation Framework for Wireless Internet Access Networks
Hyoung-Kee Choi, Jitae Shin 453

WDM: An Energy-Efficient Multi-hop Routing Algorithm for Wireless Sensor Networks
Zheng Zengwei, Wu Zhaohui, Lin Huaizhong, Zheng Kougen 461

Forwarding Scheme Extension for Fast and Secure Handoff in Hierarchical MIPv6
Hoseong Jeon, Jungmuk Lim, Hyunseung Choo, Gyung-Leen Park 468

Back-Up Chord: Chord Ring Recovery Protocol for P2P File Sharing over MANETs
Hong-Jong Jeong, Dongkyun Kim, Jeomki Song, Byung-yeub Kim, Jeong-Su Park 477

PATM: Priority-Based Adaptive Topology Management for Efficient Routing in Ad Hoc Networks
Haixia Tan, Weilin Zeng, Lichun Bao 485

Practical and Provably-Secure Multicasting over High-Delay Networks
Junghyun Nam, Hyunjue Kim, Seungjoo Kim, Dongho Won, Hyungkyu Yang 493

A Novel IDS Agent Distributing Protocol for MANETs
Jin Xin, Zhang Yao-Xue, Zhou Yue-Zhi, Wei Yaya 502

ID-Based Secure Session Key Exchange Scheme to Reduce Registration Delay with AAA in Mobile IP Networks
Kwang Cheol Jeong, Hyunseung Choo, Sang Yong Ha 510

An Efficient Wireless Resource Allocation Based on a Data Compressor Predictor
Min Zhang, Xiaolong Yang, Hong Jiang 519

A Seamless Handover Mechanism for IEEE 802.16e Broadband Wireless Access
Kyung-ah Kim, Chong-Kwon Kim, Tongsok Kim 527

Fault Tolerant Coverage Model for Sensor Networks
Doina Bein, Wolfgang W. Bein, Srilaxmi Malladi 535

Detection Algorithms Based on Chip-Level Processing for DS/CDMA Code Acquisition in Fast Fading Channels
Seokho Yoon, Jee-Hyong Lee, Sun Yong Kim 543

Clustering-Based Distributed Precomputation for Quality-of-Service Routing
Yong Cui and Jianping Wu 551

Traffic Grooming Algorithm Using Shortest EDPs Table in WDM Mesh Networks
Seungsoo Lee, Tae-Jin Lee, Min Young Chung, Hyunseung Choo 559

Efficient Indexing of Moving Objects Using Time-Based Partitioning with R-Tree
Youn Chul Jung, Hee Yong Youn, Ungmo Kim 568

Publish/Subscribe Systems on Node and Link Error Prone Mobile Environments
Sangyoon Oh, Sangmi Lee Pallickara, Sunghoon Ko, Jai-Hoon Kim, Geoffrey Fox 576

A Power Efficient Routing Protocol in Wireless Sensor Networks
Hyunsook Kim, Jungpil Ryu, Kijun Han 585

Applying Mobile Agent to Intrusion Response for Ad Hoc Networks
Ping Yi, Yiping Zhong, Shiyong Zhang 593

A Vertical Handoff Decision Process and Algorithm Based on Context Information in CDMA-WLAN Interworking
Jang-Sub Kim, Min-Young Chung, Dong-Ryeol Shin 601

Workshop on "Dynamic Data Driven Application System"

Dynamic Data Driven Applications Systems: New Capabilities for Application Simulations and Measurements
Frederica Darema 610

Dynamic Data Driven Methodologies for Multiphysics System Modeling and Simulation
J. Michopoulos, C. Farhat, E. Houstis, P. Tsompanopoulou, H. Zhang, T. Gullaud 616

Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD
Beth Plale, Dennis Gannon, Dan Reed, Sara Graves, Kelvin Droegemeier, Bob Wilhelmson, Mohan Ramamurthy 624

Towards a Dynamic Data Driven Application System for Wildfire Simulation
Jan Mandel, Lynn S. Bennethum, Mingshi Chen, Janice L. Coen, Craig C. Douglas, Leopoldo P. Franca, Craig J. Johns, Minjeong Kim, Andrew V. Knyazev, Robert Kremens, Vaibhav Kulkarni, Guan Qin, Anthony Vodacek, Jianjia Wu, Wei Zhao, Adam Zornes 632

Multiscale Interpolation, Backward in Time Error Analysis for Data-Driven Contaminant Simulation
Craig C. Douglas, Yalchin Efendiev, Richard Ewing, Victor Ginting, Raytcho Lazarov, Martin J. Cole, Greg Jones, Chris R. Johnson 640

Ensemble–Based Data Assimilation for Atmospheric Chemical Transport Models
Adrian Sandu, Emil M. Constantinescu, Wenyuan Liao, Gregory R. Carmichael, Tianfeng Chai, John H. Seinfeld, Dacian Dăescu 648

Towards Dynamic Data-Driven Optimization of Oil Well Placement
Manish Parashar, Vincent Matossian, Wolfgang Bangerth, Hector Klie, Benjamin Rutt, Tahsin Kurc, Umit Catalyurek, Joel Saltz, Mary F. Wheeler 656

High-Fidelity Simulation of Large-Scale Structures
Christoph Hoffmann, Ahmed Sameh, Ananth Grama 664

A Dynamic Data Driven Grid System for Intra-operative Image Guided Neurosurgery*
Amit Majumdar, Adam Birnbaum, Dong Ju Choi, Abhishek Trivedi, Simon K. Warfield, Kim Baldridge, Petr Krysl 672

Structure-Based Integrative Computational and Experimental Approach for the Optimization of Drug Design
Dimitrios Morikis, Christodoulos A. Floudas, John D. Lambris 680

Simulation and Visualization of Air Flow Around Bat Wings During Flight
I.V. Pivkin, E. Hueso, R. Weinstein, D.H. Laidlaw, S. Swartz, G.E. Karniadakis 689

Integrating Fire, Structure and Agent Models
A.R. Chaturvedi, S.A. Filatyev, J.P. Gore, A. Hanna, J. Means, A.K. Mellema 695

A Dynamic, Data-Driven, Decision Support System for Emergency Medical Services
Mark Gaynor, Margo Seltzer, Steve Moulton, Jim Freedman 703

Dynamic Data Driven Coupling of Continuous and Discrete Methods for 3D Tracking
Dimitris Metaxas, Gabriel Tsechpenakis 712

Semi-automated Simulation Transformation for DDDAS
David Brogan, Paul Reynolds, Robert Bartholet, Joseph Carnahan, Yannick Loitière 721

The Development of Dependable and Survivable Grids
Andrew Grimshaw, Marty Humphrey, John C. Knight, Anh Nguyen-Tuong, Jonathan Rowanhill, Glenn Wasson, Jim Basney 729

On the Fundamental Tautology of Validating Data-Driven Models and Simulations
John Michopoulos, Sam Lambrakos 738

Workshop on "Practical Aspects of High-Level Parallel Programming (PAPP)"

Managing Heterogeneity in a Grid Parallel Haskell
A. Al Zain, P.W. Trinder, H-W. Loidl, G.J. Michaelson 746

An Efficient Equi-semi-join Algorithm for Distributed Architectures
M. Bamha, G. Hains 755

Two Fundamental Concepts in Skeletal Parallel Programming
Anne Benoit, Murray Cole 764

A Formal Framework for Orthogonal Data and Control Parallelism Handling
Sonia Campa 772

Empirical Parallel Performance Prediction from Semantics-Based Profiling
Norman Scaife, Greg Michaelson, Susumu Horiguchi 781

Dynamic Memory Management in the *Loci* Framework
Yang Zhang, Edward A. Luke 790

Workshop on "New Computational Tools for Advancing Atmospheric and Oceanic Sciences"

On Adaptive Mesh Refinement for Atmospheric Pollution Models
Emil M. Constantinescu, Adrian Sandu 798

Total Energy Singular Vectors for Atmospheric Chemical Transport Models
Wenyuan Liao, Adrian Sandu 806

Application of Static Adaptive Grid Techniques for Regional-Urban Multiscale Air Quality Modeling
Daewon Byun, Peter Percell, Tanmay Basak 814

On the Accuracy of High-Order Finite Elements in Curvilinear Coordinates
Stephen J. Thomas, Amik St.-Cyr 822

Analysis of Discrete Adjoints for Upwind Numerical Schemes
Zheng Liu and Adrian Sandu 829

The Impact of Background Error on Incomplete Observations for 4D-Var Data Assimilation with the FSU GSM
I. Michael Navon, Dacian N. Daescu, Zhuo Liu 837

2005 International Workshop on Bioinformatics Research and Applications

Disjoint Segments with Maximum Density
Yen Hung Chen, Hsueh-I Lu, Chuan Yi Tang 845

Wiener Indices of Balanced Binary Trees
Sergey Bereg, Hao Wang 851

What Makes the Arc-Preserving Subsequence Problem Hard?
Guillaume Blin, Guillaume Fertin, Romeo Rizzi, Stéphane Vialette . . 860

An Efficient Dynamic Programming Algorithm and Implementation for RNA Secondary Structure Prediction
Guangming Tan, Xinchun Liu, Ninghui Sun 869

Performance Evaluation of Protein Sequence Clustering Tools
Haifeng Liu, Loo-Nin Teow 877

A Data-Adaptive Approach to cDNA Microarray Image Enhancement
Rastislav Lukac, Konstantinos N. Plataniotis, Bogdan Smolka,, Anastasios N. Venetsanopoulos 886

String Kernels of Imperfect Matches for Off-target Detection in RNA Interference
Shibin Qiu, Terran Lane 894

A New Kernel Based on High-Scored Pairs of Tri-peptides and Its Application in Prediction of Protein Subcellular Localization
Zhengdeng Lei, Yang Dai 903

Reconstructing Phylogenetic Trees of Prokaryote Genomes by Randomly Sampling Oligopeptides
Osamu Maruyama, Akiko Matsuda, Satoru Kuhara 911

Phylogenetic Networks, Trees, and Clusters
Luay Nakhleh, Li-San Wang 919

SWAT: A New Spliced Alignment Tool Tailored for Handling More Sequencing Errors
Yifeng Li, Hesham H. Ali 927

Simultaneous Alignment and Structure Prediction of RNAs
Beeta Masoumi, Marcel Turcotte 936

Clustering Using Adaptive Self-organizing Maps (ASOM) and Applications
Yong Wang, Chengyong Yang, Kalai Mathee, Giri Narasimhan 944

Experimental Analysis of a New Algorithm for Partial Haplotype Completion
Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, Lorenzo Mariani 952

Improving the Sensitivity and Specificity of Protein Homology Search by Incorporating Predicted Secondary Structures
Bin Ma, Lieyu Wu, Kaizhong Zhang 960

Profiling and Searching for RNA Pseudoknot Structures in Genomes
Chunmei Liu, Yinglei Song, Russell L. Malmberg, Liming Cai 968

Integrating Text Chunking with Mixture Hidden Markov Models for Effective Biomedical Information Extraction
Min Song, Il-Yeol Song, Xiaohua Hu, Robert B. Allen 976

k-Recombination Haplotype Inference in Pedigrees
Francis Y.L. Chin, Qiangfeng Zhang, Hong Shen 985

Improved Tag Set Design and Multiplexing Algorithms for Universal Arrays
Ion I. Măndoiu, Claudia Prăjescu, Dragoş Trincă 994

A Parallel Implementation for Determining Genomic Distances Under Deletion and Insertion
Vijaya Smitha Kolli, Hui Liu, Michelle Hong Pan, Yi Pan 1003

Phasing and Missing Data Recovery in Family Trios
Dumitru Brinza, Jingwu He, Weidong Mao, Alexander Zelikovsky .. 1011

Highly Scalable Algorithms for Robust String Barcoding
B. DasGupta, K.M. Konwar, I.I. Măndoiu, A.A. Shvartsman 1020

Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection
Ferdinando Cicalese, Peter Damaschke, Ugo Vaccaro 1029

Virtual Gene: A Gene Selection Algorithm for Sample Classification on Microarray Datasets
Xian Xu, Aidong Zhang 1038

Workshop on "Programming Grids and Metacomputing System – PGaMS2005"

Bulk Synchronous Parallel ML: Modular Implementation and Performance Prediction
Frédéric Loulergue, Frédéric Gava, David Billiet 1046

Fast Expression Templates
Jochen Härdtlein, Alexander Linke, Christoph Pflaum 1055

Solving Coupled Geoscience Problems on High Performance Computing Platforms
Dany Kemmler, Panagiotis Adamidis, Wenqing Wang, Sebastian Bauer, Olaf Kolditz 1064

H2O Metacomputing - Jini Lookup and Discovery
Dirk Gorissen, Gunther Stuer, Kurt Vanmechelen, Jan Broeckhove 1072

User Experiences with Nuclear Physics Calculations on a H2O Metacomputing System and on the BEgrid
P. Hellinckx, K. Vanmechelen, G. Stuer, F. Arickx, J. Broeckhove . . 1080

Author Index 1089

Teaching High-Performance Computing on a High-Performance Cluster

Martin Bernreuther, Markus Brenk, Hans-Joachim Bungartz, Ralf-Peter Mundani, and Ioan Lucian Muntean

IPVS, Universität Stuttgart, Universitätsstraße 38, D-70569 Stuttgart, Germany
bungartz@ipvs.uni-stuttgart.de

Abstract. The university education in parallel and high-performance computing often suffers from a significant gap between the effects and potential performance taught in the lectures on the one hand and those practically experienced in exercises or lab courses on the other hand. With a small number of processors, the results obtained are often hardly convincing; however, supercomputers are rarely accessible to students doing their first steps in parallel programming. In this contribution, we present our experiences of how a state-of-the-art mid-size Linux cluster, bought and operated on a department level primarily for education and algorithm development purposes, can be used for teaching a large variety of HPC aspects. Special focus is put on the effects of such an approach on the intensity and sustainability of learning.

1 Introduction

The education in high-performance computing (HPC) at an academic level has to contend with several difficulties, the first being a disciplinary one. Within a math environment, the corresponding courses (if existing at all) are typically not that different from standard numerical courses; in computer science study programs, the focus is often restricted to the architecture of supercomputers and to their programming; from the point of view of a field of application, finally, HPC is frequently limited to speeding up some specific code as much as possible. Really interdisciplinary courses for mixed target groups are still rather rare. In addition to the standard fears of contact, curricular issues or different educational backgrounds may hinder the implementation of appropriate HPC courses.

A second and perhaps even more serious problem is the frequent lack of accessibility of suitable computers. Typical courses in parallel computing, e.g., focus on algorithmic and complexity issues, and they do this from a more or less theoretical point of view. If there is an accompanying practical (i.e. programming) part, students have to program small toy problems of toy size in MPI or OpenMP. One reason is that, in most cases, only a network of work stations just combined ad hoc via standard Ethernet connections, a department server (perhaps some smaller shared memory machine), or possibly a small experimental Linux cluster are accessible to students. The big machines in the computing

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1–9, 2005.
© Springer-Verlag Berlin Heidelberg 2005

centres, however, where tasks of a more realistic size could be tackled and where real and significant experiences could be gathered, are out of reach. They are reserved for research projects, not for teaching; they are needed for production runs and must not be wasted for the first trial and error steps of a class of twenty students running twenty bug-ridden programs for the same tasks.

Hence, university teachers often are confronted with an, at least at first glance, astonishingly low enthusiasm on the students' side – which actually is, of course, far from being astonishing: sorting ten thousand numbers in 0.5 seconds instead of 1.8 seconds can hardly be expected to be considered as a scientific breakthrough. And a fictitious upscaling of both problem sizes and peak performance may be a rational argument helpful in a lecture, but it won't be sufficient to infect even motivated students by the professor's enthusiasm.

What are the consequences? On the one hand, a parallel computer to be used in teaching should be sufficiently large to also allow for large-scale runs; the number of processors should be big enough to allow for reasonable studies on parallel performance, i.e. speed-up, parallel efficiency, scaled problem analysis, and so on. Briefly, all effects that can be observed on the really big machines should be visible here as well. On the other hand, of course, for being really accessible even to beginners, it must be installed and operated locally, i.e. in the respective department and not in some computing centre, where always service aspects are dominating. Consequently, we need an affordable solution which does not require a lot of staff for operation.

Against this background, two institutes of the Universität Stuttgart – the Institute of Applied Analysis and Numerical Simulation located in the math and physics faculty and the Institute of Parallel and Distributed Systems located in the computer science, electrical engineering, and information technology faculty – decided to invest in a mid-size cluster slightly below Top500 performance in the list of June 2004. The, probably, somewhat unusual aspect is that this cluster is primarily used for teaching and education, ranging from beginners' courses up to PhD studies. Since its installation in March 2004, our cluster *Mozart* [9] has been used in a large variety of courses and projects [8], many of which have been remodelled and tailored to *Mozart*. It is the intention of this contribution to discuss the possibilities and chances of such a kind of HPC education.

The remainder of this paper is organized as follows: In Sect. 2, we briefly present the cluster *Mozart* and its main features. Then, in Sects. 3 and 4, the types of courses involved and some representative exercises are described. In Sect. 5, we discuss the expected and the actually observed outcomes for our participating students. Finally, some concluding remarks close the discussion.

2 The Cluster Mozart

Mozart [9] is a standard mid-size homogeneous Linux cluster which was completely delivered and installed as a whole by the German company MEGWARE [2]. *Mozart* consists of 64 dual-Xeon computing nodes and of one additional

dual-Xeon node playing the part of the master and front-end node. In brief, the node configuration is as follows: a supermicro X5DPA-GG or X5DPL-8GM (master) ATX chipset; two Intel Xeon 3.06 GHz FSB 533 1 MB cache CPUs; 4 GB DDR-RAM; finally, an IBM 180GXP/60 GB or Ultrastar 146Z10/73 GB (master) hard disk drive. The cluster's theoretical peak performance resulting from this hardware configuration is close to 785 GFlops. As interconnect technology, *Mozart* has an InfiniBand 4x network (8 Gbit/s, 72-port Mellanox Gazelle Switch). Additionally, for administrative tasks, there is a Gigabit Ethernet interconnect (1 Gbit/s, three manageable HP ProCurve 4824 24-port switches and one HP ProCurve 2708 switch) in use. The system's overall cost was about 390.000 Euro.

Currently, *Mozart* is run with the Redhat 9.0 Linux operating system. For administration and monitoring purposes, Clustware v2.0.13 and a couple of other tools are used. For parallelisation, the MPI implementations MPICH-1.2.5.2 and MVAPICH-0.9.2 with InfiniBand support [10] are available. As compilers, there are the GNU gcc 3.x compiler and the current Intel compilers for C/C++ and FORTRAN with OpenMP support. To evaluate *Mozart's* potential, the Linpack benchmark runs were done in April 2004, showing a maximum sustained performance of about 597 GFlops or 76% of the theoretical peak performance. Figure 1 illustrates the Linpack performance for various numbers of processors.

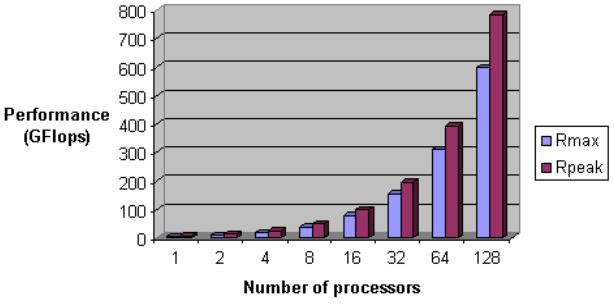


Fig. 1. Peak performance (R_{peak}) and maximum sustained performance ($R_{\max}$) of the HPL benchmark on *Mozart* for various numbers of processors

At the moment, besides studies of parallel computing itself and besides the teaching activities mentioned in the following sections, most of *Mozart's* remaining CPU time is used for finite element algorithm development and for simulation projects on flow simulations, fluid-structure interactions, molecular dynamics simulations, and traffic simulations.

3 Types of Courses Involved

At present, *Mozart* is used for a large variety of teaching activities and types of courses at graduate level, mainly within the diploma programs *Mathematics*,

Computer Science, and *Software Engineering*, and within the master program *Information Technology*.

First, there was a seminar *Cluster Computing* tailored to *Mozart*. Starting from a list of topics such as *Processor and Board Alternatives*, *Parallel I/O*, *State-of-the-art Network Technologies*, *MPI*, *Programming Models and Environments*, *Tuning Distributed Applications*, *Performance Evaluation and Benchmarking*, *Dynamic Load Balancing*, or *From Cluster to Grid Computing*, each student had to choose a topic and to prepare both a lecture and a paper on it. Wherever it was useful, students got accounts to work on *Mozart* and to use their results for their presentations. In the course evaluation, all participants showed enthusiastic about such an opportunity in a (typically more theoretically oriented) format such as a seminar, and quite a lot of them used their seminar participation as an entry point to some more practical project work on the cluster.

Second, we use *Mozart* as a platform for the practical exercises in the *Parallel Programming* course, where more or less standard problems for first parallel experiences are to be solved. Since programming plays only a minor part here, and since the programs to be written are, typically, rather short and simple, a four- or eight-node machine, e.g., would be sufficient, too. Nevertheless, the bigger cluster's function as a motivator and appetizer should not be neglected.

These first steps are then continued and extended in the lab course *Parallel and Distributed Programming*, where students shall get a deeper insight into MPI and OpenMP programming as two crucial fundamentals of HPC [1, 5, 6, 7]. We use OpenMP for both fine-grain parallelism on the two processors of each of *Mozart's* nodes and programming available shared-memory machines. During the first two thirds of the semester, students program parallel algorithms for classical problems from computer science such as sorting, coding and decoding, or solving linear systems of equations. In the last third, each group of 2-3 students selects one project which shall lead to an (at least to some extent) bigger piece of parallel software and which has some more realistic and, possibly, HPC background. We will discuss a few of these project tasks in the next section. It shall also be mentioned that we offer another lab course *Scientific Computing* that emphasizes the numerical aspects of HPC, where *Mozart* has also been integrated.

Furthermore, *Mozart* is a main pillar in our student project *Computational Steering*. Student projects are a specific format in the graduate part of the *Software Engineering* diploma program. Here, for one year, a group of 8-10 students work together on some larger software engineering project, under conditions which have been designed to be as realistic as possible. For example, one assistant plays the customer part, specifying the final product at the beginning and controlling progress throughout the year, but without being involved in the actual work; another assistant works as the adviser. Both project and configuration management have to be implemented, and students have to organize all aspects of teamwork and time management by themselves. In particular, the *Computational Steering* student project aims at getting a virtual wind tunnel software system, where numerical flow simulations and the visualization of the resulting flow fields are combined in an interactive VR environment in order to

control, foster, and accelerate the design process of a car, for example. For all numerical computations, *Mozart* is the target platform. This student project will also be discussed a bit more in the next section. Additionally, of course, a lot of individual projects such as diploma theses are running in the context of *Mozart*.

Finally, our group has several cooperations with Eastern European universities. For example, three students from the NTU Donezk, Ukraine, and fourteen students from different universities of the Balkan States have spent a 2-6-month internship at our department during the last three years. The main objective is to provide some insight into modern parallel computing and simulation scenarios. Recently, these internships have involved *Mozart*, too. For example, in the summer of 2004, one PhD student of computer science from Cluj-Napoca, Romania, studied aspects of Quality of Service over InfiniBand networks, and at the moment, one Ukrainian student is working on program development tools for *Mozart*. For both of them, this is the first contact with a parallel computer of reasonable size.

It is important to note that, formerly, we also integrated the use of parallel computers into our HPC-related teaching, of course. These were a small 8-processor cluster, our 8-processor (shared memory) department server, and various machines available at our university's computing centre (who also operates one of Germany's four federal supercomputing centres). However, the use and the outcomes were very much restricted: On our own machines, access was easy, but the learning goals were hard to achieve with eight processors; in the computing centre, the size was no problem, but the accessibility was limited (batch mode with long turnaround times, the need of writing proposals before getting accounts, or even no access for classes).

4 Typical Tasks and Exercises

In this section, we present some representative tasks from the lab course *Parallel and Distributed Programming* and the student project *Computational Steering*.

4.1 Projects in the Lab Course *Parallel Programming*

With their respective project, students finish the lab course *Parallel Programming*. All projects last for roughly six weeks, and they shall deal with some scalable parallel application, either programmed in OpenMP or MPI, where at least a few typical challenges have to be encountered.

One such project is to program a parallel simulator of the game "Scotland Yard", where a group of players tries to track down the mysterious Mr. X who moves around in London using buses, taxis, or the subway. Various algorithmic variants have to be implemented and to be analysed statistically in order to improve the chances of either Mr. X or of his pursuers. With sufficient computing resources, the resolution of the field can be increased, making the game more realistic and the job more complex from the computational point of view.

Another project, which is a bit closer to simulation and, hence, standard HPC applications, deals with the implementation of a parallel traffic simulator based on a cellular automaton algorithmic approach as suggested in [3]. To be precise, a parallel simulator for the simple scenario of car traffic on a one-lane highway has to be written. For parallelisation purposes, the highway is subdivided into an appropriate number of overlapping sections, each section corresponding to one parallel process. The basic underlying algorithm is rather simple. Each cell may contain at most one car at a time. The different states of a cell describe either the state of being empty or the speed of the respective car. This speed changes depending on the distance to the next car driving ahead, typically with some probabilistic component being included. There are quite a lot of interesting issues to be considered: How large should the regions of overlap be chosen? When is the optimum starting point of communication during a simulation step? How should an efficient data transfer be organized? How can a both simple and efficient dynamic load balancing be designed, which turns out to be necessary if the single nodes are not dedicated for the traffic simulation, but have to serve other parallel applications, too? For analysing communication, tools such as profilers are to be used. Finally, to prepare a further upscaling of the application (for the simulation of larger traffic networks, for example) and, hence, the extension to grid computing, the necessary changes in organizing the data transfer due to a now strongly varying communication bandwidth between the nodes have to be considered and implemented.

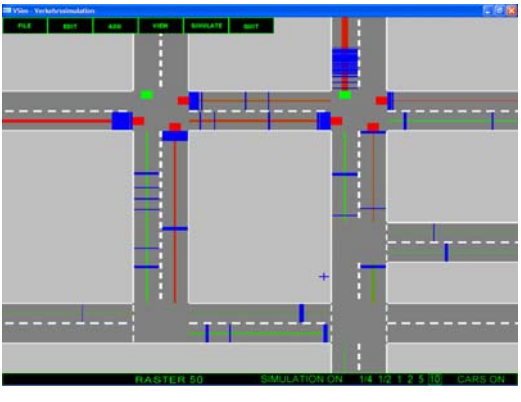


Fig. 2. Visualization of a basic traffic simulator

4.2 Tasks in the Student Project *Computational Steering*

The main idea behind the development of a virtual wind tunnel is to change the classical workflow of the design process. Preprocessing, simulation, and postprocessing phases merge to a simulation steering process illustrated in Fig. 3. There are several consequences, especially for the simulation part. We need a reactive software immediately handling user requests such as model changes or results. Since speed is a crucial prerequisite, efficient algorithms are necessary, as well

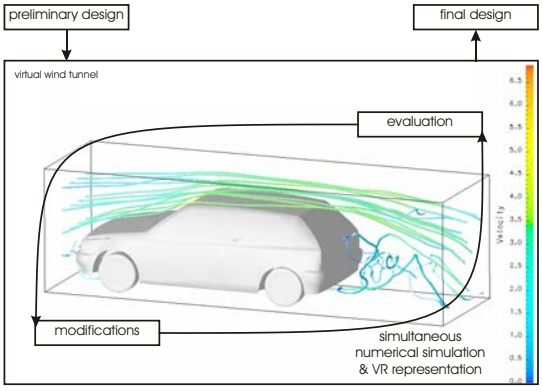


Fig. 3. Simulation steering workflow

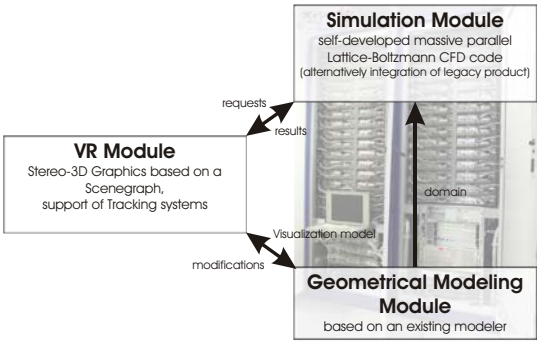


Fig. 4. Software modules

as a powerful hardware platform like *Mozart*. The machine is needed exclusively during the whole run and is busy doing online calculations.

From the technical point of view, the project consists of the three modules shown in Fig. 4, which form a distributed system: the *Geometric Modelling Module*, the *Simulation Module*, and the *Virtual Reality Module*. A common technique for representing solid objects is the boundary representation based on free form surfaces. In this project, we base the modelling on OpenCASCADE [4], a freely available open source development platform for 3D modelling applications. Surface meshing generates the visualization model, which is transfered, as well as the control points, to the VR module. The simulation module also derives its domain from this solid model. Next, as a CFD approach suitable for the steering context and for a massively parallel target system, the Lattice-Boltzmann method was chosen. A voxelization process generates the domain from the geometric model. There's also an interface to the commercial flow solver StarCD as an alternative. Finally, the user shall interact with the software system in a VR environment, in our case a power wall. Modifications are done with the help

of the control points through a flystick, and the resulting changes are submitted to the geometric modelling module. At the moment, the VR part runs on a SGI Onyx; in the future, this job shall be done by a cluster, too. The direct access to this hardware is an indispensable prerequisite for the realization of this project, which is still under development.

5 Expected and Experienced Outcomes

Among all positive effects that we expected and, actually, can observe, the following three are probably the most important. First, the direct integration of a moderate supercomputer into undergraduate and graduate teaching allows students to get a much more integral way of looking at scientific computing and HPC, letting them experience the effects of underlying models, numerical algorithms, and implementation issues on parallel performance in a very natural way. Second, several properties of parallel algorithms such as communication behaviour or scalability become obvious only with a larger number of processors. Thus, the possibility of running experiments with up to 128 processors improves students' practical experiences with parallel and supercomputer programming, as well as their intuitive ability of evaluating parallel algorithms. Finally, the variety of starting points for improving parallel performance such as algorithmic variants, compiler support, alternative parallelisation strategies, general tuning mechanisms, and so on can be explored in a by far more intense way.

6 Concluding Remarks

In this paper, we have discussed various ways how a modern mid-size Linux cluster can be used in university education in parallel computing or HPC. Although most of the exercises introduced could also be done on smaller parallel machines, on simple Ethernet-based networks of workstations, or on the supercomputer(s) available in the local computing centre, our experiences show that the educational outcomes and the learning effects are clearly improved. Furthermore, an increased motivation of students to further dive into HPC can be observed.

References

1. A. Grama et al. *Introduction to Parallel Computing.* Addison-Wesley, 2003.
2. MEGWARE Computers, Chemnitz, Germany. www.megware.com.
3. K. Nagel and M. Schreckenberg. *A cellular automaton model for freeway traffic.* J. Phys. I France **2** (1992), pp. 2221-2229.
4. *OpenCASCADE* www.opencascade.org/.
5. P. S. Pacheco. *Parallel Programming with MPI.* Morgan Kaufmann, 1997.
6. M. Quinn. *Parallel Programming in C with MPI and OpenMP.* Internat. ed., McGraw-Hill, New York, 2003.

7. M. Snir et al. *MPI: The Complete Reference (vol. 1 and 2)*. MIT Press, 1998.
8. Universität Stuttgart, IPVS. *Courses offered by the simulation department*. www.ipvs.uni-stuttgart.de/abteilungen/sgs/lehre/lehrveranstaltungen/start/en.
9. Universität Stuttgart, IPVS. *The Linux Cluster Mozart*. www.ipvs.uni-stuttgart.de/abteilungen/sgs/abteilung/ausstattung/mozart/.
10. *MPI over InfiniBand Project*. nowlab.cis.ohio-state.edu/projects/mpi-iba/.

Teaching High Performance Computing Parallelizing a Real Computational Science Application

Giovanni Aloisio, Massimo Cafaro, Italo Epicoco, and Gianvito Quarta

Center for Advanced Computational Technologies,
University of Lecce/ISUFI, Italy
{giovanni.aloisio, massimo.cafaro,
italo.epicoco, gianvito.quarta}@unile.it

Abstract. In this paper we present our approach to teaching High Performance Computing at both the undergraduate and graduate level. For undergraduate students, we emphasize the key role of an hands on approach. Parallel computing theory at this stage is kept at minimal level since this knowledge is fundamental, but our main goal for undergraduate students is the required ability to develop real parallel applications. For this reason we spend about one third of the class lectures on the theory and remaining two thirds on programming environments, tools and libraries for development of parallel applications. The availability of widely adopted standards provides us, as teachers of high performance computing, with the opportunity to present parallel algorithms uniformly, to teach how portable parallel software must be developed, how to use parallel libraries etc. When teaching at the graduate level instead, we spend more time on theory, highlighting all of the relevant aspects of parallel computation, models, parallel complexity classes, architectures, message passing and shared memory paradigms etc. In particular, we stress the key points of design and analysis of parallel applications. As a case study, we present to our students the parallelization of a real computational science application, namely a remote sensing SAR (Synthetic Aperture Radar) processor, using both MPI and OpenMP.

1 Introduction

Introducing parallel computing in the undergraduate curriculum provides current students with the knowledge they will certainly need in the years to come. For undergraduate students, we emphasize the key role of an hands on approach. The study program provides students with a degree in Computer Engineering; the program can be considered at the bachelor level. We refer to just one course of the undergraduate program in this paper. We do also have master level courses (Parallel Computing I and Parallel Computing II) and Ph.D. level courses. In the undergraduate program parallel computing theory is kept at minimal level since this knowledge is fundamental, but our main goal for undergraduate students is the required ability to develop real parallel applications. For this reason

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 10–17, 2005.
© Springer-Verlag Berlin Heidelberg 2005

we spend about one third of the class lectures on the theory and remaining two thirds on programming environments, tools and libraries for development of parallel applications.

At the undergraduate level we simply introduce briefly the need for parallelism, the concepts of speedup, efficiency and scalability, and the models undelying message passing and shared memory programming. We rely on Foster's PCAM design methodology [1] when designing message passing applications, and on dependency analysis of loop variables for shared memory programming.

Performance analysis include Amdahl [2] and Gustafson-Barsis [3] laws, the Karp-Flatt metric [4] and iso-efficiency analysis. The availability of widely adopted standards provides us, as teachers of high performance computing, with the opportunity to present parallel algorithms uniformly, to teach how portable parallel software must be developed, how to use parallel libraries etc. We utilize both MPI OpenMP. The course introduces the most important functionalities available in the MPI 1.2 specification, and all of the OpenMP library. The main programming language is C.

Each student is required to parallelize, as a short project, a real application. We have found that assigning projects to groups of students does not work as expected. We thought that organizing students in groups would have fostered the key concept of collaboration, and provided fertile ground for discussions etc. This proved to be true for graduate students, whilst for undergraduates the net effect was that only one or two students per group actually did the job assigned. Therefore, we require that undergraduate students carry out individually their projects. The project is not necessarily done during the course: each student is required to present his project when actually taking the course examination (which can also happen several months after the end of the course, since we do have ten examination per year for each course). Thus, a student may work on his/her project as much as he/she needs. Likewise, we do allow up to one year for the final bachelor thesis (this differs from many universities both in Europe and USA, but is quite common in Italy); in turn we usually get very satisfactory results.

This paper presents the parallelization made by one of our undergraduate students of a real computational science application, namely a remote sensing SAR [5] raw data processor, using both MPI and OpenMP. SAR processing [6] applies signal processing to produce a high resolution image from SAR raw data. High image precision leads to more complicated algorithms and higher computing time; in contrast, space agencies often have real-time or near real-time requirements. As matter of fact, SAR processing algorithms are computationally intensive and require fast access to secondary storage. In order to accelerate the processing, SAR focusing has been implemented on special purpose architectures and on HPC platforms. Nevertheless, special purpose architectures have relatively high cost, when compared to HPC platforms that are now becoming increasingly popular for this task. The paper is organized as follows. Section 2 recalls the SAR processor application and the rules of the parallelization contest we organized. Section 3 describes the winning parallel SAR processor and Section 4 concludes the paper.

2 SAR Image Processing

The SAR sensor is installed on a satellite or aircraft that flies at constant altitude. SAR works transmitting a beam of electromagnetic (EM) radiation in the microwave region of the EM spectrum. The back scattered earths radiation is intercepted by the SAR antenna and recorded. The received echoes are digitalized and stored in memory as a two dimensional array of samples. One dimension of the array represents the distance in the slant range direction between the sensor and the target and it is referred to as the range direction. The other dimension represents the along-track or azimuth direction.

The main goal of SAR processing is to reconstruct the scene from all of the pulses reflected by each single target. In essence, it can be considered as a two dimensional focusing operation. The first, relatively straightforward, is range focusing; it requires the de-chirping of the received echoes. Azimuth focusing depends upon the Doppler histories produced by each point in the target field and it is similar to the de-chirping operation used in the range direction. This is complicated however by the fact that these Doppler histories are range dependent, so azimuth compression must have the same range dependency. It is also necessary to correct the data in order to account for sensor motion and Earth rotation.

SAR focusing has been implemented, generally, using the classic range-Doppler algorithm [7] or chirp-scaling algorithm [8]. The range-Doppler algorithm does first range compression operation and then azimuth compression. During azimuth processing, a space-variant interpolation is required to compensate the migration of signal energy through range resolution cells. In general, interpolation may require significant computational time.

The AESAR package, a sequential range-Doppler SAR image processor developed by the Italian Space Agency, has been selected for our last year parallellization contest. The contest rules for undergraduate students were: (i) students can freely decide how to parallelize the code, (ii) modifications to the legacy code must be kept at a minimum due to engineering costs, and the target architecture is an HP AlphaServer SC machine. This machine is a cluster of SMP nodes, and each node contains four alpha processors. For graduate students the target machine was an HP RX6000, a cluster of Itanium 2 nodes, each node containing two processors, and the code could be refactored and reengineered as needed. We describe now the chosen computational science application and how the sequential range-Doppler algorithm works. This is is the most widely used algorithm for SAR focusing. We describe first the sequential algorithm. The core steps of range-Doppler algorithm follow.

After raw data have been read, the image frame is divided into blocks, overlapped in azimuth direction. Then, a Fast Fourier Transform (FFT) is performed in the range direction; subsequently range compression is performed through a complex multiplication of the range lines with a range reference function. The range reference function is obtained from Doppler rate, extracted by a parameter file. Finally, an IFFT (Inverse FFT) is performed in the range direction. Before azimuth FFT, the corner turning operation must be performed. It con-

sists of a transposition of the memory arrangement of 2-dimensional array of data. Then, the FFT in azimuth direction is performed, followed by range cell migration correction which requires a shift and interpolation operation. The azimuth compression requires a complex multiplication of the azimuth column by the azimuth reference function. The azimuth reference function is calculated for each azimuth column, using the Doppler centroid value estimated before. Finally, an IFFT in azimuth direction is performed to complete the focusing process.

3 Parallel SAR Processor

After a careful analysis of the sequential algorithm, the student decided to instrument and profile code execution in order to determine computationally intensive numerical kernels. He found that the majority of the time is spent on Range and Azimuth Compression. According to the range-Doppler algorithm, the student then proposed an hybrid parallelization approach. Course grain parallelism for this application entails distributing the image frame segments to MPI processes. The entire raw image frame is divided into a fixed number of segments, and for each segment range and azimuth compression is computed sequentially. The segments are independent of each other and partly overlapped as needed by the focusing algorithm. The size of the overlap region is imposed by phisical constraint on the processing.

Fine grain parallelism, usually not suitable for MPI applications, is instead effective using OpenMP. Therefore, our student parallelization strategy distributes the lines belonging to a given segment to available threads. Given a segment, both range and azimuth compression are computed in parallel, one after the other. The hybrid MPI/OpenMP approach takes advantage of the benefits of both message passing and shared memory models, and makes better use of the proposed architecture, a cluster of SMP nodes. Indeed, since the number of segments is fixed, so is the number of MPI processes. In such a situation, requiring a specific number of processes severely limits scalability. Instead, the simultaneous use of OpenMP allows exploiting additional CPUs: the natural MPI domain decomposition strategy for the application can still be used, running the required number of MPI processes, and OpenMP threads can be used to further distribute the work among threads.

The frame-level parallelization has been implemented using MPI. To optimize the performance, the student made segment computation independent from other segments. Indeed, he tried first sending the overlapped lines needed by a segment computation to the process in charge of that segment. Even though the communication network was a Quadrics QS-Net, he found that for this application and target machine it is best to avoid inter-node communication. This of course leads to an implementation that includes redundant computation: to process each segment independently from the others, it is necessary that each process is also responsible for the rows in the overlap region. Then, for the MPI implementation there is no communication overhead.

The image segmentation mechanism must satisfy the following requirement: the size of segments, must be greater than the number of overlapped lines, because this is the length of the filter used to process raw data in azimuth direction. Moreover, a bigger segment size implies reduced performances due to the FFT routines. This leads to a total of nine segments. The constraint on the number of segments entails that when the number of MPI processes does not divide evenly the number of the segments, the computational load is not balanced properly and so the parallel algorithm should include a load balancing strategy.

The segment-level parallelization model has been implemented using OpenMP. The student correctly identified and removed loop carried dependencies in order to parallelize loops. In order to achieve better performances, the student tried to minimize parallel overhead. The main issue and source of overhead is the presence of critical sections, where multiple threads potentially can modify shared variables. The student minimized this overhead partially rewriting the sequential code so that each thread, when possible, has its own copy of the variables, even though the approach entails the use of additional space. Other factors that contribute to parallel overhead are: (i) parallel directives used to execute parallel loops, (ii) the loop scheduling to balance the computational load and the atomic construct used to provide exclusive access to variables being updated; (iii) accesses to different locations in the same cache line (set of entries in a single cache location). The former two sources of overhead increase linearly with the number of threads involved. The latter depends on the number of threads that read and/or write different locations in the same cache line and on the amount of data assigned to each thread.

3.1 Parallel Model

Here we describe the student model for this application that predicts the parallel time when using p MPI processes and t OpenMP threads. Given

- n number of segments;
- r total number of rows;
- c total number of columns;
- o number of overlapped rows between contiguous segments;
- T_i time spent for data inizialization, Doppler evaluation;
- T_{ec} time spent for echo correction for one row;
- T_{r_conv} time spent to compute the convolution between one row and chirp signal;
- T_{a_conv} time spent to compute the convolution between one column and estimated chirp signal along range direction;
- T_{rcm} time spent for range cell migration correction for one azimuth column;
- T_{file} time spent to write a line to file.

$$T(p,t) = T_i + \left\lceil \frac{n}{p} \right\rceil (T_{range} + T_{azimuth}) \tag{1}$$

where T_{range} is defined by:

$$T_{range} = (T_{ec} + T_{r_conv})(\frac{r}{n} + o)\frac{1}{t} \quad (2)$$

and $T_{azimuth}$ is

$$T_{azimuth} = (T_{rcm} + T_{a_conv} + T_{file})\frac{c}{t} \quad (3)$$

These parameters have been evaluated profiling the application. The sequential code exploited the traditional Cooley-Tukey FFT algorithm. The student was aware, due to class lectures, that better alternatives exist. He substituted the FFT calls with the corresponding functions from the FFTW library [9] and estimated that performances are better for 4096 complex elements. Considering this, he fixed the number of segments (nine).

The model has been validated against experimental runs of the application in order to assess its ability to predict the parallel time, and related measures such as speedup and efficiency, as shown in Figures 1, 2 and 3. The application was run varying the number of MPI processes from one to three, and the number of threads per process from one to four, since the parallel queue available to students on the target machine is made of three SMP nodes, each one containg four CPUs. As shown in Figure 1, the model correctly approximates the parallel execution time; in particular, the slightly superlinear speedup obatined when using a single MPI process and a varying number of OpenMP threads, up to four, is due to cache effects. Finally, when using two MPI processes and four OpenMP threads, for a total of eight CPUs, we observe a decrease of efficiency. This is expected, since in this case the computational load is not perfectly balanced because one process is responsible for five segments, whilst the other gets the remaining four segments.

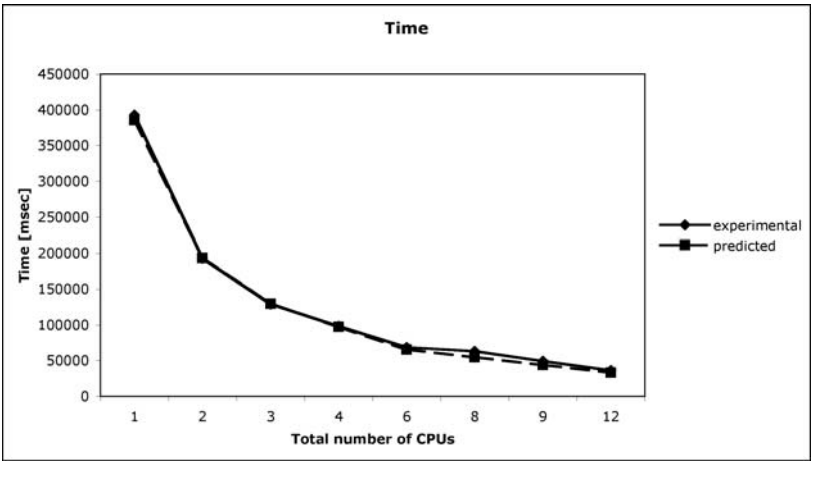


Fig. 1. Parallel Time

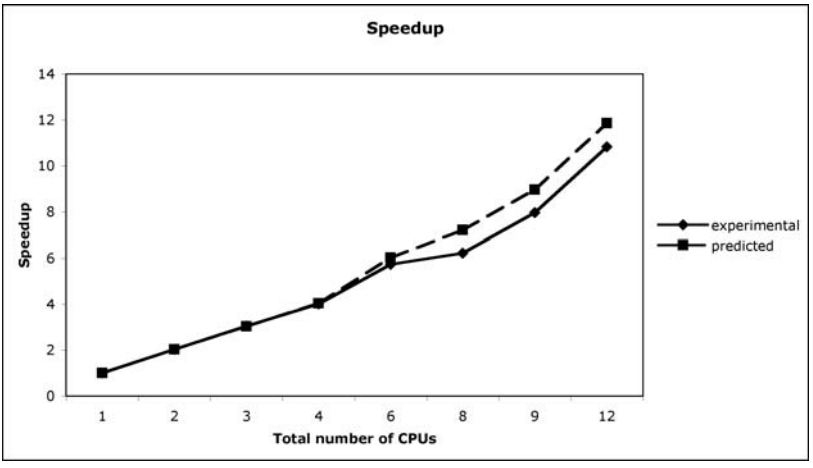


Fig. 2. Speedup

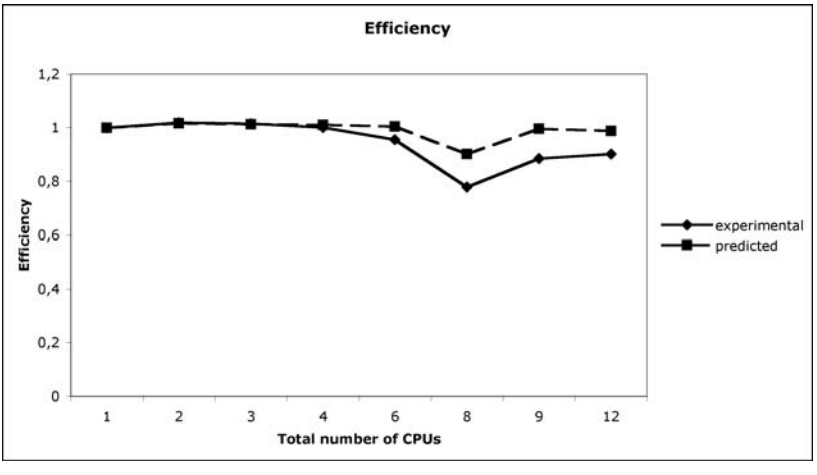


Fig. 3. Efficiency

4 Conclusions

In this paper we have described the parallelization of a real computational science application, SAR processing, reporting the experience of an undergraduate student parallelizing a range-Doppler legacy code using an hybrid MPI/OpenMp approach. When the students are given enough time, the experience reported in this paper is a good representative of average outcomes for this HPC course. The one-student team approach was feasible because students had enough time (several months if needed) to complete their homework project. And it was interesting to see that students did not require too much help from teachers/assistants.

Moreover, cooperation was explicitly forbidden during the project: there is no point in having one student teams if students can collaborate. However, exchange of experience is always beneficial and we do allow this during the course. We have found that, besides teaching traditional examples of parallel applications such as matrix multiplication etc, students like the hands on approach we use in our Paralleling Computing course. The parallel contest we organize as part of the course proves to be extremely useful, especially for undergraduate students to better understand parallel computing theory and related practical issues. The student was able to parallelize the proposed application and to correctly model its parallel execution time, thus meeting the main goals of the course.

References

1. Foster I.: Designing and Building Parallel Programs, Addison-Wesley, 1995
2. Amdahl G: Validity of the single processor approach to achieving large scale computing capabilities, Proc. AFIPS, Vol. 30, pp. 483–485, 1967
3. Gustafson, J. L.: Reevaluating Amdahl's law, Communications of the ACM 31(5), pp. 532–533, 1988
4. Karp A. H., Flatt H. P.: Measuring parallel processor performance, Communications of the ACM 33(5), pp. 539–543, 1990
5. Elachi, C.: Spaceborne Radar Remote Sensing: Applications and Techniques, IEEE Press, 1988
6. Barber B.C.: Theory of digital imaging from orbital synthetic-aperture radar, INT. J. Remote Sensing, 6, 1009, 1985
7. Smith A. M.: A new apporach to range-Doppler SAR processing, Journal Remote Sensing, 1991 VOL. 12, NO 2, 235-251
8. Raney R.K., Runge H., Bamler R., Cumming I.G., Wong F.H.: Precision SAR Processing Using Chirp Scaling. IEEE Transactions on Geoscience and Remote Sensing, 32(4):786-799, July 1994
9. Frigo M., Johnson S. G.: FFTW: An Adaptive Software Architecture for the FFT. ICASSP conference proceedings 1998 vol. 3, pp. 1381-1384

Introducing Design Patterns, Graphical User Interfaces and Threads Within the Context of a High Performance Computing Application

James Roper and Alistair P. Rendell

Department of Computer Science, Australian National University,
Canberra ACT0200, Australia
alistair.rendell@anu.edu.au

Abstract. The cross fertilization of methods and techniques between different subject areas in the undergraduate curriculum is a challenge, especially at the more advanced levels. This paper describes an attempt to achieve this through a tutorial based around a traditional high performance computing application, namely molecular dynamics. The tutorial exposes students to elements of software design patterns, the construction of graphical user interfaces, and concurrent programming concepts. The tutorial targets senior undergraduate or early postgraduate students and is relevant to both those majoring in computing as well as other science disciplines.

1 Introduction

By its very nature computational science is interdisciplinary requiring mathematical, computing and application specific skills. At most tertiary institutions, however, the undergraduate curriculum funnels students towards specialization. Accepting this raises the obvious question of how existing courses can be modified to show students that skills learnt in one domain may be applied to another, or alternatively, how techniques developed in another area might be useful in their field of study.

The above divide is particularly noticeable between students majoring in some aspect of computing versus those majoring in other science subjects – like chemistry, physics or mathematics. Thus while the chemistry, physics or mathematics student may take a class or two in computer science during their freshman year, timetable constraints and pre-requisite requirements often inhibit them from taking higher level computer science classes. Likewise a student pursuing a computer science major may get some exposure to first year chemistry, physics or mathematics, but rarely do they progress to higher level courses. While some knowledge of a discipline at first year is useful, computer science freshman courses generally teach little more than basic programming, with more advanced concepts like software design and analysis or concurrent programming left to later years.

Noting the above, and as part of the computational science initiative of the Australian Partnership in Advanced Computing [1] we have been working to design a series of tutorials that can be used by senior students in both computer science and

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 18 – 26, 2005.
© Springer-Verlag Berlin Heidelberg 2005

other science courses. The aim is to construct a tutorial that, depending on the situation, can be used to expose computer science majors to aspects of scientific programming, or introduce aspects of computer science to students with other science majors. This paper outlines our work on one such tutorial, where the objective is to illustrate the use of design patterns, graphical user interfaces and threading in the setting of a traditional high performance computing application code. Molecular dynamics was chosen as the application domain since it is relatively easy to comprehend with little science or mathematics background.

2 Tutorial Background

The tutorial assumes that the reader is a competent programmer, but does not assume that they have experience in any particular programming paradigm, thus concepts like inheritance and threading are described. Visual Python (VPython) [2] was chosen as the programming language. This is a 3-D graphics system that is an extension of the Python programming language and has the advantage that the programmer need not worry about the underlying mechanism of how to build the display environment, but is just required to specify the layout of the shapes used in any particular model. (A subsequent tutorial, currently under development, will include mixed programming using VPython and C/C++.)

While the tutorial does assume a working knowledge of VPython, when this is lacking students are referred to two earlier tutorials [3] that were developed at the Australian National University and begin by modeling a "bouncing ball" confined within a box, and then evolve into a simulation of hard sphere gas particles. These earlier tutorials assume little or no programming experience and lead the user through the process of creating a ball (atom), setting it in motion, confining it to a box, and eventually to performing a gas simulation and comparing the simulated velocity distributed with the expected Maxwellian distribution. These earlier tutorials have frequently been given to students in grades 11 or 12 and their teachers with great success, demonstrating the ease of programming with Python and the interactivity imparted by using VPython to display results.

The molecular dynamics tutorial is divided into 5 modules. The first 4 are substantive and are designed to introduce molecular dynamics, design patterns, graphical user interfaces, and threading respectively. The final module serves largely to review outcomes from the previous modules, and present a much more advanced final product with a discussion of the additional considerations used in producing this final product. Below we briefly summarize the key features of the 5 modules.

2.1 Module #1 - Basic Molecular Dynamics

The aim of module 1 is to obtain a basic working molecular dynamics code. The starting point is a cubic "box of atoms" of size R and with N atoms positioned along each axis – giving a total of N^3 atoms. At this point the "box" is used primarily to provide a simple definition for the starting coordinates of each atom, but in due course it is linked to the idea of atom packing and concepts like body centered cubic or face

centered cubic structures. Integrated into the initial problem definition section of this module is a discussion of data structures, and a comparison of using a list of atom objects versus a list of vectors. The ease of interpretation associated with a list of atom objects is contrasted with the performance advantage of a list of vectors.

The tutorial then introduces an elementary Lennard-Jones interaction potential and leads the students through evaluation of the total potential energy of the system and force on each particle. With these components two options are pursued, one to minimize the structure of the system, and another to perform a dynamics calculation using a simple integrator and recording the potential, kinetic and total energy at each timestep. With these basic pieces in place the students are invited to add the few lines of VPython required to visualize the system as the coordinates change.

At this stage the students have already produced a simple molecular dynamics code and are in a position to explore some the underlying issues involved in using the code. For example they experimentally evaluate the performance of their code with respect to the various input parameters, such as timestep size, number of timesteps, number of atoms, initial atom coordinates. The behavior of the code as a function of the timestep is also studied, with the goal that the student recognizes when the results produced are no longer physically valid. The form of the interaction potential is considered, with the idea of producing an $O(n)$ scaling algorithm by using a cutoff to only compute numerical significant interactions investigated.

2.2 Module #2 - An Introduction to Software Design

At the end of module 1 the student has produced a very simple program capable of performing basic structural minimizations and dynamic simulations on a group of interacting atoms. Module 2 posses the question of adding functionality to their code, e.g. what if we want to add an option to colour atoms according to energy, or augment the graphics to display arrows on each atom indicating the force on that atom. The aim here is to indicate to the student that without any formal methods of design it is possible that their software can very quickly become hard to develop and manage. The students are pointed to an article by Dianna Mullet [4] highlighting “The Software Crisis”.

Following this preamble the student is introduced to the concept of a design pattern. Patterns are an attempt to describe workable solutions to known problems in a manner that enables these solutions to be easily applied when new “similar” problems arise. Although the concept of “patterns” and “pattern languages” dates back to the late 70’s when they were introduced by Alexander in the context of building design [5], it was not until the late 80’s that widespread interest was aroused within the computer science community. Since then the benefits of this approach to software design has been widely recognized, in part due to the landmark book “Design Patterns: Elements of Reusable Object-Oriented Software”, that was published by Gamma, Helm, Johnson and Vlissides in 1995 [6] and lists 23 software design patterns.

While the concept and use of design patterns is now well established within the computer science and software engineering community, this is not the case within the

high performance computing community. The aim of this module is to make the reader aware of design patterns and give them a flavor for their utility in a computational science setting. It is not intended to be a comprehensive introduction to the topic. The tutorial starts by considering the model-view-controller pattern as this is one of the most commonly used design patterns in graphical applications. It separates the software into three main areas.

The model: this is the part of the software that performs the function that the software is being written for. In the case of this tutorial there are two models, a minimizer and a dynamics simulator, and the algorithms that go with them. The model provides an interface for accessing the data that it processes, and for telling it how to behave.

The View: this takes the data that the model has calculated and renders it in a format suitable for a person or another program. The format may be displaying it as graphical text, storing it on disk or transmitting it over the internet. In this tutorial the view is the VPython output.

The Controller: this tells the view anything it needs to know about how to display the model, or the model anything it needs to know about how to behave. It starts and ends the program. The controller will usually handle all user input, and send it to the models or views accordingly.

Within this environment if the user wants to print out the kinetic, potential and total energies they would create a new view, but they would only have to write this code once since both the minimizer and simulator can use the same view.

A problem with the model-view-controller pattern is dealing with more views. Each time a new view is added code would need to be added to the models to tell them to update the new view; if the user wanted to stop updating a particular view, conditional statements need to be added to enable this. A better approach is provided by the observer pattern, this completely decouples the view from the models. For the purpose of the observer pattern the models are the subjects and the views the observers. The observers are purely that, they observe a subject and don't have any impact on the subject. To facilitate this a common interface to all observers is required, then subjects can easily notify them when something has changed.

Since the observers are now responsible for knowing which subject to look at, two subject methods are required:

`Attach(observer)`: adds an observer to the list of observers

`Detach(observer)`: removes an observer from the list of observers

As well as this a new method "`notify()`" is required to inform all observers that the subject has changed. With this framework it then becomes the responsibility of the controller to create the observer(s) and appropriate subject, attach observer(s) to the subject, and finally tell the subject to run.

Given a basic understanding of the observer pattern, its implementation in the context of the molecular dynamics code is discussed. This involves defining abstract base classes for the subject and observer, with two derived subject classes for the

minimizer and simulator, and one derived observer class for the renderer. As this requires the use of inheritance, the tutorial contains a brief overview of this concept and its implementation in Python. Figure 1 contains a unified modeling language (UML) diagram showing the observer pattern as applied to the molecular dynamics code.

A variety of exercises are included to illustrate the advantages of the new modular software design. For example, the user is invited to implement another observer that simply prints the values of the total energies in a text window – a task that can now be done without making any changes to the simulator or minimizer.

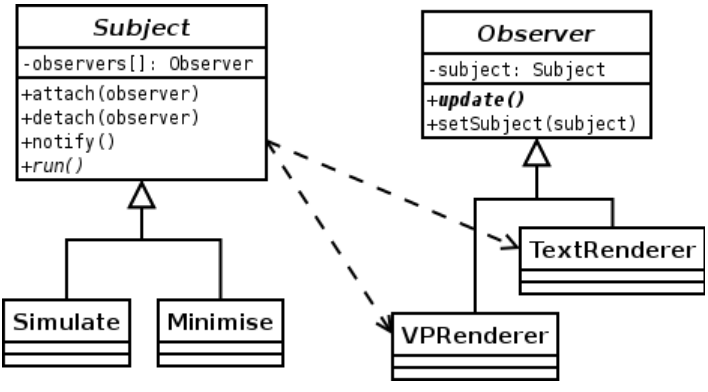


Fig. 1. The parent subject class contains all the code that handles the observers. The children, Simulate and Minimise, implement the `run()` routine, and whenever their state changes, they call `notify()`, which in turn calls update() on each observer that is currently attached to that subject. The `update()` routine will then use the publicly available data (such as the lists of atoms or forces on each atom) provided by Simulate or Minimise and render it appropriately

2.3 Module #3 – Graphical User Interface

At the end of module 2 the basic molecular dynamics code from module 1 has been rewritten to be based on a modular object oriented design conforming to the observer design pattern. Input to the code is still provided via the command line, which with only a few parameters is not a major issue. The student is, however, asked to consider how expanding the number of input parameters soon leads to unwieldy command line input, and perhaps a better option is to use a Graphical User Interface (GUI).

In designing the GUI portion of the tutorial three possible graphics packages were considered:

Visual Python: this has its own interface package. It is very easy to use, with a simple call used to check if the user has interacted with it. Unfortunately, its functionality is somewhat lacking, in particular the only way to input a value is through the use of a slider and the user has no means of knowing the exact value. For the purpose of our simulation this was unsatisfactory

Tkinter: is the main graphics package used in Python applications [7]. It is a set of Python bindings for the Tcl/Tk graphics package. It is part of any Python distribution and as a result is easily portable

PyGTK: is a set of Python binding for GTK [8]. GTK (The Gimp ToolKit) is a relatively easy to use graphics package that was originally written for and used by the GNU Image Manipulation Program (GIMP). It has since become one of the most widely used graphics packages on UNIX platforms.

Since GTK has many more widgets than Tcl/Tk, is also faster, and since we had some previous experience with GTK this was chosen for use in the tutorial. Students who have never used any graphics package before are recommended to do the "Getting Started" chapter of the PyGTK tutorial [8].

The initial goal for the tutorial is to create a basic GUI containing a drop down menu that can select either "Minimize" or "Simulate", and four boxes where values for the number of atoms along each side of the cube, length of each side, timestep and total number of timesteps can be given. At the bottom of the GUI there are two buttons, one to start and one to quit the calculation.

Placement of the main GTK loop within the controller is discussed, as is the difference between the graphics that the VPython renderer displays and the graphics that the controller displays. Exercises are included that have the student modifying the GUI so that when the minimizer is selected the input options for the timestep size and total number of timesteps are disabled (or "grayed out").

2.4 Module #4 – Threading

At the end of Module 3 the students have not only produced a well designed molecular dynamics application code, but have also constructed a basic graphical user interface. Their attention is now drawn to the fact that there is no way of stopping or suspending the minimization/simulation once the start button has been depressed and until it has finished minimizing or run for the required number of timesteps. What if we wanted to change some parameters on the fly, such as to turn text rendering on or off, or to add arrows to the graphical output indicating the forces on the atoms? The difficulty is due to the need to run two loops at once, one is the GTK loop which waits for a widget to emit a signal and then calls any attached callbacks, while the other is the main simulation loop that updates the atomic positions. The easiest way to run both these loops concurrently is to have two threads.

The tutorial uses the Python threading module to implement the subject (i.e. either the minimizer and simulator) as a separate thread from the master thread. Sharing of data between threads is discussed, as are basic synchronization constructs and concepts like "busy wait". The student is assigned the task of adding a pause button to the controller, modifying the start button to read either start or stop, and making the program automatically desensitize the pause button when the simulation has stopped.

2.5 Module #5: The Final Product

By the end of module 4 the student has assembled quite a sophisticated computational science application code and environment. The purpose of final module is to present

the student with a more advanced version compared to their final product and to provide some discussion as to how this differs from their code. In the provided code the controller is packed full with additional features, there are more functions, and more design patterns have been applied. A screenshot of the final product is given in Figure 2.

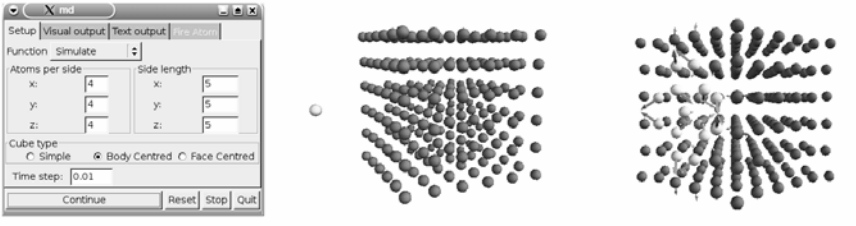


Fig. 2. Control panel and two screen shots to illustrate i) the firing of an atom into a previously minimized cube of target atoms and ii) the resulting force vectors after collision

Some of the additional issues raised include:

Template pattern: what if we wanted to place a charge on the atoms, or define periodic boundary conditions, or in some other way modify the basic interaction between the particles in our system? This would require a modification to each subject, and then the creation of multiple sets of subjects. A better option is to use a template pattern, e.g. MDTemplate, that has several deferred methods, including force evaluation, potential evaluation and velocity update. For each type of function a class can be created that inherits from MDTemplate. To use this new implementation a method is added to the subject class, so that the subject class can transparently use whatever utility it is given.

Decorator Pattern: the provided code includes a new subject called "fire" that shoots an atom into the main cube of atoms (such as might be done in a simulation of atomic deposition). The fire subject is essentially identical to the original simulation subject, the only difference being that the fire subject adds an atom with a velocity vector that ensures that it will collide with the cube of atoms. Since fire and simulate are so similar it is more efficient to let fire inherit from simulate and then redefine or add methods as required. The end result is that fire only has the code relevant to adding a firing atom, and it does not need to worry about simulating it. The only other required modification is to the renderer that needs to be changed in order to accommodate the target and arrows. Due, however, to its well planned design none of its code needs to be altered, only the relevant parts added.

A Configuration Class: the provided code can minimize a cube of atoms and then fire an atom at the resulting structure. To enable this functionality a new "configuration" class was added to store all the information concerning the

structure of the atoms. It is initially imported into each subject before running and then exported when that subject finishes.

The GUI: the final GUI is considerably more complex than at the end of module 4. As the size and complexity of the application increases it is very easy for the code associated with the GUI to become very messy. Some discussion on how to improve the design of the GUI is given.

At the end of module 1 the students had a VPython code of roughly 200 lines. The code provided in module 5 and excluding the GTK interface contains roughly 650 lines. This is much larger than the original code, but it is now possible to write quickly a controller that will do far more than the original program, and do this without touching the exiting code. It is also possible to add new functions and incorporate different environments and forces in a relatively clean and easy manner.

3 Conclusions

In the 70's and 80's computation was successfully applied to modeling a range of physical phenomena. While much useful work was undertaken to develop the underlying methods and algorithms the associated programs typically evolved in a rather haphazard fashion, had primitive input/output capabilities, and were run on large mainframe systems with little interactivity. Today we seek to apply computational methods to much more complex problems, using computer systems that are considerably more advanced. Graphical user interfaces are considered the norm, greatly helping in input preparation and post simulation analysis. As a consequence computational science requires practitioners with some understanding of good software design, how to build and use a graphical user interface, and an appreciation for concurrent and parallel programming issues. In this tutorial we have attempted to demonstrate the importance of these skills in the context of building quite a sophisticated molecular simulation environment. VPython was found to be a useful vehicle for conveying these ideas.

Acknowledgements

The authors gratefully acknowledge support from the Computational Science Education Program of the Australian Partnership in Advanced Computing.

References

1. The Australian Partnership in Advanced Computing, see http://www.apac.edu.au
2. Visual Python, see http://www.vpython.org
3. S. Roberts, H. Gardner, S. Press, L. Stals, "Teaching Computational Science Uisng VPython and Virtual Reality", Lecture notes in computer science, **3039**, 1218-1225 (2004).
4. D. Mullet, "The Software Crisis", see http://www.unt.edu/benchmarks/archives /1999/july99/crisis.htm

5. C. Alexander, S. Ishikawa, and M. Silverstein, “A Pattern Language: Towns, Buildings, Construction”, Oxford University Press, New York (1977) ISBN 0195019199
6. E. Gamma, R. Helm, R. Johnson and J. Vlissides, “Design Patterns: Elements of Reusable Object Oriented Software”, Addison-Wesley (1995) ISBN 0201633612
7. Tk Interface (Tkinter), see http://docs.python.org/lib/module-Tkinter.html
8. PyGTK, see http://www.pygtk.org/

High Performance Computing Education for Students in Computational Engineering

Uwe Fabricius, Christoph Freundl, Harald Köstler, and Ulrich Rüde

Lehrstuhl für Simulation, Institut für Informatik,
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
{Uwe.Fabricius, Christoph.Freundl, Harald.Koestler,
Ulrich.Ruede}@cs.fau.de
http://www10.informatik.uni-erlangen.de/

Abstract. Numerical simulation using high performance computing has become a key technology for many scientific disciplines. Consequently, high performance computing courses constitute an essential component within the undergraduate and graduate programs in Computational Engineering at University of Erlangen-Nuremberg. These courses are also offered as optional courses in other degree programs, such as for majors in computer science.

1 The Erlangen Computational Engineering Program

The courses in high performance computing at University of Erlangen-Nuremberg are primarily motivated by the the *Computational Engineering* (CE) program that has been initiated by the Department of Computer Science in 1997 as a prototype two-year postgraduate program leading to a Masterdegree. The corresponding undergraduate program has been started in 1999. Together these two programs are accepting approximately 30 new undergraduate students and 45 graduate students, annually.

The traditional German university degree in the sciences and the engineering disciplines is the *Diplom* which corresponds approximately to the academic level of a Masterdegree in the US educational system. Currently the system is being reformed according to the so-called *Bologna Process*, a political agenda that is aimed at introducing a Europe-wide, standardized university degree system by 2010. This reform process will lead to an educational structure with a first degree on the Bachelorlevel, on top of which graduate programs leading to the Masterand Doctorate can be built.

The Erlangen Computational Engineering programs are prototype implementations of this new system, since they already award Bachelorand Masterdegrees. Generally, the Bachelor-Master structure of academic programs is still in an experimental stage in Germany, but the transition away from the Diplom degree will accelerate during the next couple of years. All core courses of the CE Masterprogram are taught in English and are thus open to international students without knowledge of German.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 27–35, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The CE program is built around a core of computer science and mathematics courses. Additionally, each student must select a *technical application* field. Currently CE in Erlangen offers application specializations in

- Mechanical Engineering
- Micro Electronics
- Information Technology
- Automatic Control
- Thermo- and Fluid Dynamics
- Material Sciences
- Sensor Technology

The curriculum requires approximately an equal number of credits in mathematics, computer science, and the application field. The university education system in Germany traditionally puts a strong emphasis on thesis work and thus, like the Diplom degree, the Masterrequires a full six month thesis, and even for the Bachelordegree students must spend three months on thesis work. A more detailed description of the programs can be found in [ER-CSE]. Up-to-date information can be obtained from the Internet[1].

2 Bavarian Graduate School in Computational Engineering

Starting in fall 2004, the Bavarian Graduate School In Computational Engineering[2] (BGSCE) has been established as a network of excellence between Friedrich-Alexander-Universitt Erlangen (FAU) and Technische Universität München (TUM). The partners in this consortium consist of three existing MasterPrograms in the field of Computational Science and Engineering:

- Computational Mechanics (COME)[3] at TUM
- Computational Science and Engineering (CSE)[4] at TUM
- Computational Engineering (CE) at FAU

Students of the Bavarian Graduate School in Computational Engineering are recruited from the best students of each participating Masterprogram. These students stay enrolled in their home program, but they are required to take an extra load of 30 ECTS[5] credits. In turn they are awarded a Masterdegree *with Honours*. The extra credits must been earned partly in classes out of the other Masterprograms. This is made possible by offering suitable courses in block form or in the form of summer schools.

[1] http://www10.informatik.uni-erlangen.de/CE/
[2] http://www.bgsce.de/
[3] http://www.come.tum.de/
[4] http://www.cse.tum.de/
[5] European Credit Transfer System
http://europa.eu.int/comm/education/programmes/socrates/ects_en.html

This trans-institutional network of excellence has won special funding in a state-wide competition from the state of Bavaria in its Elite-Network[6] initiative.

3 Simulation as a Core Field in Computational Engineering

The undergraduate CE program is based on the traditional German four-semester *engineering mathematics* sequence, but this is complemented by two semesters of numerical mathematics in the third and fourth semester. Additionally, students are required to take a newly developed course in *Algorithms and Data Structures for Continuous Systems* to be taken in the fourth semester. This course is unique in that it presents algorithms for handling continuous data, such as required for image and video processing, computer graphics, visualization, and the simulation of technical systems. It contains material from each of these fields together with their theoretical background in (numerical) mathematics.

Building on the material taught in these courses during the first two years of study, the Department of Computer Science offers a two semester sequence in *Simulation and Scientific Computing* (SISC). These courses are designed to provide a broad view of numerical simulation and as such they put a significant emphasis on the basic elements of *High Performance Computing* (HPC). The SISC sequence is required for CE students and can be chosen as optional courses within the Computer Science (CS) curriculum. New Masterdegree students who do not yet have an equivalent background are also required to take the SISC sequence.

Besides the core curriculum of required courses, students can and must select additional credit hours from an exhaustive list of optional courses that can be taken either from the student's application field, computer science, or applied mathematics. Though any course of the conventional degree programs of the participating departments can be chosen, students are intensively advised and guided individually to enable them to find suitable combinations of courses.

Among the optional courses there are several that offer a further specialization in high performance computing topics. The most prominent here are *Parallel Algorithms* and *Programming Techniques for Supercomputers*. The structure outlined here has been the result of an update of the curriculum in 2003/04 and as such differs slightly from the state as described in [ER-CSE, RR].

On the graduate and advanced undergraduate level, we have also created new classes with the goal to better bridge the gap between the disciplines. These courses are interdisciplinary and are taught jointly by faculty from the different departments. They often integrate an aspect of high performance computing.

One such course is *Numerical Simulation of Fluids* which is presented jointly by Chemical Engineering and Computer Science faculty. Using [NFL] as the basic text, the course teaches students to develop an incompressible Navier-Stokes solver from scratch. This is a significant difference from how computational fluid

[6] http://www.elitenetzwerk-bayern.de/en/index.html

dynamics is usually being taught in engineering. While a classical course would introduce students to existing computational fluid dynamics software and teach them how to use (and possibly extend) it, our course is deliberately designed to teach the fundamentals of flow simulation, even if this comes at the cost of being restricted to what students can accomplish in programming during one semester. The first half of the course has weekly assignments that result in a core 2D fluid simulator. The method is based on a staggered grid finite difference discretization, explicit time stepping for the velocities, and a marker-and-cell method for dealing with nontrivial geometries.

From our experience, the feeling of accomplishment results in a very high motivation for the students. When the core solver has been implemented, students are individually guided to adapt and apply their code to a more complicated application scenario. For this, they form teams of up to three students. Typical projects include the parallelization of the code either for distributed or shared memory parallel execution. In this way the course teaches high performance computing aspects in an integrated form, as driven by a typical application scenario. For students this is especially profitable, when they additionally take one of the special courses with HPC focus and as outlined in the following section.

4 High Performance Computing Courses

The department offers several courses with special emphasis on high performance computing. Besides the mandatory material included in the two-semester sequence *Simulation and Scientific Computing* (SISC), Computational Engineering students can choose courses from the following list

- Cluster Computing
- Parallel Algorithms
- Programming Techniques for Supercomputers.

While the first is being designed primarily for computer science students, giving an overview of parallel computing with clusters from a CS perspective, the latter two are primarily oriented at the needs of CE students.

Parallel Algorithms provides a general overview of parallel computing techniques and algorithms. This is complemented by *Programming Techniques for Supercomputers* which is taught out of the computing center and is specifically aimed at the performance optimization and parallelization of typical simulation algorithms.

Each of the courses is self contained so that there is some unavoidable overlap in the material presented, when a student takes all courses, but this makes it possible to choose these courses independently. Typically, a student will choose two of these courses depending on his or her special interests.

5 High Performance Computing Topics

In the following we will describe in some more detail some of the material that is currently taught as part of the course SISC and which is mandatory for all CE students.

The development over the past decade has brought enormous progress in the performance of computers. Unfortunately, the performance boost has come at the price of an ever increasing complexity of systems, an increasing internal parallelism even within the CPU, deep pipelines, and a widening gap between memory and CPU performance. Consequently, it becomes increasingly difficult to exploit the performance potential even of single CPU systems. Additionally, many applications require parallel processing using clusters of PCs or parallel supercomputers.

Some of the program optimization techniques are similar to the problem of vectorizing algorithms, other aspects are typical for hierarchical memory systems, other are particular for specific CPU families. Outside the high performance community this problem receives relatively little attention and is therefore not well taught in standard computer science classes. However, since this knowledge has become essential for successful high performance computing, it should be addressed in the basic HPC classes of a CE program.

Modern techniques of single CPU program optimization are therefore included in a sequence of seven 90 minute lecture units within the SISC sequence. The material is partly based on the monograph by Goedecker and Hoisie [GH] which is used as the textbook for this part of the course. The material is roughly organized into the units

- review of computer architecture
- examples of high performance systems
- basic efficiency guidelines
- code profiling and tuning
- optimization of floating point operations
- optimizing of memory access
- cache blocking

Students are grouped in teams of three and have to work on three assignments which have the character of a little projects. The results have to be presented by the team in a short 10 minute talks. Each team is required to prepare a set of approximately 10 slides. The presentations (also of the German students) are given in English, thus providing students with a valuable experience in giving presentations to an international audience. This scheme has evolved over several years and has proved to be very motivating for all students. In particular it often leads a very fruitful combination of cooperation and competition. Generally, students often put much more than average effort into the assignments and often go much beyond the required work. The topics of the three core assignments may change from year to year. A typical setup is as follows:

- Matrix-matrix multiply: Here students are given the (seemingly) simple task to code a matrix-matrix multiplication and to compare the performance of

different loop orders. Students are then given access to a highly optimized code that performs about ten times faster. This comparison code is taken from the ATLAS web site[7], see also [ATLAS]. Students are required to use profiling tools (as discussed in the lectures) and present their analysis why the differences in performance occur. Though this is not required in the assignment, the best students will typically explore blocking techniques or discuss using the Strassen multiplication algorithm for faster performance.
- Gauss-Seidel-Iteration for Poisson's equation in 2-D on a square grid in red-black order, as used as a smoother within a multigrid algorithm: Students are required to use blocking techniques, experiment with different CPU architectures, analyze and explain their findings.
- Gauss-Seidel for a (stationary) variable coefficient Poisson-like partial differential equation (PDE), implemented both on a structured grid and alternatively using a compressed row sparse matrix data structure. Students are required to try various optimization techniques and to analyze and present their performance measurements, in particular in comparison with the previous assignment.

The remainder of the lecture covers special algorithms. Typical topics are the Lattice Boltzmann method for simulating fluid flow and as an example of using cellular automata in CE. Other topics include the conjugate gradient and multigrid algorithms as the dominant algorithms in PDE solvers. Typical further assignments will require students to implement these algorithms, and thus students will have ample opportunity to exercise the HPC programming techniques.

Parallel programming is another topic that will be included in elementary form within SISC, once the above mentioned curriculum change has propagated accordingly. More information on the current contents of the course can be downloaded from its web site[8].

The audience in SISC is mixed, consisting of both CE students and CS students. CE students are usually primarily motivated by the applications they want to study, e.g. in fluid dynamics or electrical engineering. For these students, HPC techniques are therefore a tool necessary to successfully solve application problems. CE students who come out of our own undergraduate program have a quite substantial programming expertise and background knowledge in computer architecture. Since they have a good grasp of their application field and the basic algorithms, they tend to be well prepared for the class.

The situation is different for CE Masterstudents whose basic education is in an engineering field, and who often do not have a systematic CS education. For these students, much of the material related to computer and systems architecture is new and requires substantial effort to master. For these students, the assignments are especially valuable, since this may be their first in-depth programming experience.

[7] http://math-atlas.sourceforge.net/
[8] http://www10.informatik.uni-erlangen.de/de/Teaching/Courses/SiwiR/

CS students tend to have the opposite problem. Naturally they are more intrigued by the aspects of HPC to computer architecture, compilers, and programming methodology, but they often do not have a good background in the applications and algorithms. The algorithms taught in the typical CS curriculum do not emphasize numerical simulation, and furthermore, the mathematics in the CS curriculum does not go deep enough in calculus and numerical analysis, as would be required for understanding the more complex mathematical algorithms. From another perspective: this is exactly, why a separate CE program is necessary besides the standard CS curriculum.

The heterogeneous mix of the audience makes teaching the SISC sequence a challenge, but often it is exactly this diversity of student backgrounds and the large variety of interests that leads to an especially lively discussion among the students. The student presentations have proved to be an effective scheme in promoting this kind of interdisciplinary exchange. Furthermore, we believe that this is an essential aspect of Computational Science and Engineering itself, and that exposing students to the need of collaborating and discussing scientific problems with students from a different background is an important part of their education.

While SISC is designed to cover the basic elements of HPC programming, the elective courses and in particular *Programming Techniques for Supercomputers* extend the material to in-depth parallel computing using MPI and OpenMP. Though this course can also be taken by itself, it is a natural follow-up for those CE or CS students who want to put a special focus on high performance computing as part of their education.

6 Parallel High Performance Computers

Exposing students to current HPC systems is essential for a competitive CE education.

For the SISC course described above, students are given access to various workstations and PCs. Currently this is typically an up-to-date Pentium 4 based system and some Athlon-based systems, all running Linux. Previously the class was using several Alpha-based systems (under True-64 Unix) that are still available but which are by now somewhat outdated. An Opteron-based system with dual and quad nodes with a total of 60 processors has been newly acquired in Nov 2004 and is being made available to the students in the course[9]. For students, this system is usually only available with some restrictions, however, it is fully available for student projects, Bachelor-, Master-, and PhD thesis research.

All machines are accessible from the course laboratory, but most students prefer to work remotely either from home or from other labs on campus. Though this may be personally convenient, some students deliberately choose to work in the lab, since this makes it easier to work as a team, exchange ideas with other

[9] http://www10.informatik.uni-erlangen.de/Cluster/hpc.shtml

students, and provides the opportunity to receive individual help and advise from the tutors.

For higher performance requirements (in particular for the course *Programming Techniques for Supercomputers*) the larger machines within the Erlangen Computing Center are available. This currently includes an 300 processor Intel IA-32 based cluster, an SGI Origin 3400 with 28 MIPS R14000 processors and 56 GByte memory, plus an SGI Altix 3700 super-cluster with 28 Itanium2 CPUs and 112 GByte memory.

Additionally, the University of Erlangen is part of a consortium in High Performance Computing (KONWIHR)[10] operated by the state of Bavaria and has access to the supercomputers at the Leibniz computing center[11] of the Bavarian Academy of Sciences. The largest machine there is currently a Two-Teraflop Hitachi SR-8000-F1 super computer with 1300 CPUs. This machine was originally installed in year 2000, and is now scheduled for replacement with a 60 TFlop supercomputer in 2006.

Machines of this class are usually not available freely to students in the above courses, but will be made available for thesis research on the Bachelor, Master, or PhD level, or to students working as research assistants in projects using these machines. The declared goal of the above courses is to train students to become competent users of such HPC computers and thus enable them to work at the leading edge in CE research.

Additionally, the CE program is involved in several international collaborations within Europe and the USA, through which we can gain access (usually for benchmarking comparisons) to an even wider class of machines. Currently the primary machine class not directly available to us are classical vector supercomputers, since our own Fujitsu based vector computer was outdated and has been taken offline, recently. Access to NEC vector supercomputers is e.g. possible through the Stuttgart supercomputer center.

7 Conclusions

At University of Erlangen we have established a systematic set of HPC classes which are primarily oriented at the requirements of our new Computational Engineering program, but which are also open to students in other degree programs. For the courses and thesis research, a comprehensive selection of up-to-date HPC-systems is available.

References

[NFL] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer: *Numerical Simulation in Fluid Dynamics: A Practical Introduction*, SIAM, 1997.

[10] http://konwihr.in.tum.de/index_e.html
[11] http://www.lrz-muenchen.de/wir/intro/en/#super

[GH] Stefan Goedecker and Adolfy Hoisie: *Performance Optimization of Numerically Intensive Codes*, SIAM, 2001.

[ER-CSE] U. Ruede: *Computational Engineering Programs at the University of Erlangen-Nuremberg*, in Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III, P.M.A. Sloot, C.J. Kenneth Tan, J.J. Dongarra, A.G. Hoekstra (Eds.), Lecture Notes in Computer Science 2331, pp. 852–860, Springer, 2002.

[RR] Rosemary A. Renaut and Ulrich Ruede: Editorial, Future Gener. Comput. Syst. 19, vol. 8, p.1265, Elsevier, 2003.

[ATLAS] R. Clint Whaley and Antoine Petitet and Jack J. Dongarra: *Automated Empirical Optimization of Software and the ATLAS Project*, Parallel Computing 27, 1–2, pp 3–35, 2001.

Integrating Teaching and Research in HPC: Experiences and Opportunities

M. Berzins[1], R.M. Kirby[1], and C.R. Johnson[1]

School of Computing and Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA

Abstract. Multidisciplinary research reliant upon high-performance computing stretches the traditional educational framework into which it is often shoehorned. Multidisciplinary research centers, coupled with flexible and responsive educational plans, provide a means of training the next generation of multidisciplinary computational scientists and engineers. The purpose of this paper is to address some of the issues associated with providing appropriate education for those being trained by, and in the future being employed by, multidisciplinary computational science research environments.

1 Introduction

The emerging multidisciplinary area of Computing, as distinguished from traditional Computer Science, is the study and solution of a new class of multidisciplinary problems whose solution depends on the combination of state-of-the-art computer science coupled with domain-specific expertise in such areas as medicine, engineering, biology, and geophysics. In a Computing Research Association article [1], Foley describes Computing as the integration of Computer Science and other disciplines to address problems of wide interest as illustrated in Figure 1. Multidisciplinary Computing is one of the fastest growing research areas in the US and Europe.

Examples of typical multidisciplinary computing problems are:

- How can we efficiently store, model, visualize and understand the mass of data generated by the human genome program?
- How might we model, simulate and visualize the functions of the heart and brain to better diagnose and treat cardiac and neural abnormalities with a view to improving the quality of life?
- How might we compute solutions to realistic physical models of dangerous situations such as explosions with a view to improving safety?

The next wave of industry growth will focus on opportunities resulting from the answers to questions such as these. Examples of Computing efforts at the University of Utah include the School of Computing, Scientific Computing and Imaging (SCI) Institute, the Department of Energy (DOE) ASCI Center for

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 36–43, 2005.
© Springer-Verlag Berlin Heidelberg 2005

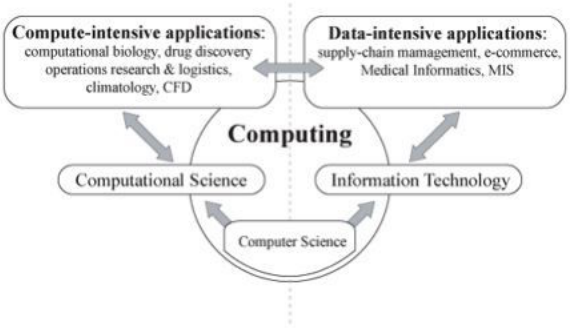


Fig. 1. Relationships between Computing, Computer Science and Applications, adapted from J. Foley's CRA article [1]

the Simulation of Accidental Fires and Explosions (C-SAFE), the NSF Grid-Computing Environment for Research, and Education in Computational Engineering Science, among several others.

The objective of this paper is to present an educational model that bridges the research mission of university computing research activities with the educational mission of the university in a synergistic way that benefits both the university and the student. We present a new University of Utah program that provides educational opportunities specifically enhanced by interaction with on-campus computing research activities. This program is a Ph.D. program in Computing with emphasis tracks in Scientific Computing, Computer Graphics and Visualization, and Robotics, offered through the School of Computing. It is worth stressing that these are not developments in isolation. In 1998, 31 graduate programs in computational science at U.S. Universities had been created. As of 2003, the number had grown to 47. In addition, since 1998, 16 new undergraduate degree programs in computational science had been created. The Computing track in Scientific Computing benefits from, and builds upon, the current M.S. degree program in Computational Engineering and Science (CES) [2].

The paper is organized as follows. In Section 2, we will present the research missions and research results from two large computing research centers that reside on the University of Utah campus. In Section 3, we will present details concerning the new Computing graduate degree program, with specific emphasis on how this educational programs provide a win-win situation for both the research missions of the centers and the educational mission of the university. We will use an example from a high performance computing course to illustrate the intertwined nature of classroom education and research education. We conclude in Section 4 with a summary and discussion of our findings concerning this integrated approach.

2 Multidisciplinary Research Efforts at Utah

To accurately understand and appreciate the environment in which these programs were developed, we will present a discussion of two current research centers at the University of Utah. The first of these is the Center for the Simulation of Accidental Fires and Explosions (C-SAFE), funded by the U.S. Department of Energy, which represents a center whose focus is the physical sciences and engineering. The second is the Center for Bioelectric Field Modeling, Simulation, and Visualization funded by the U.S. National Institutes of Health (NIH), which represents a center whose focus is in biomedicine and bioengineering. These two centers represent research efforts rich in opportunity for integrating teaching and research in high-performance computing.

2.1 Center for the Simulation of Accidental Fires and Explosions (C-SAFE)

C-SAFE is funded under the Department of Energy's Accelerated Strategic Computing Initiative (ASCI) program. The primary goal of C-SAFE focuses specifically on providing state-of-the-art, science-based tools for the numerical simulation of accidental fires and explosions, especially within the context of handling and storage of highly flammable materials. In Figure 2 (left) we present a visualization of a fire calculation which required the efforts of computational scientists, mathematicians and engineers. The objective of C-SAFE is to provide a system comprising a problem-solving environment (the Uintah PSE) [3, 4] in which fundamental chemistry and engineering physics are fully coupled with non-linear solvers, optimization, computational steering, visualization and experimental data verification.

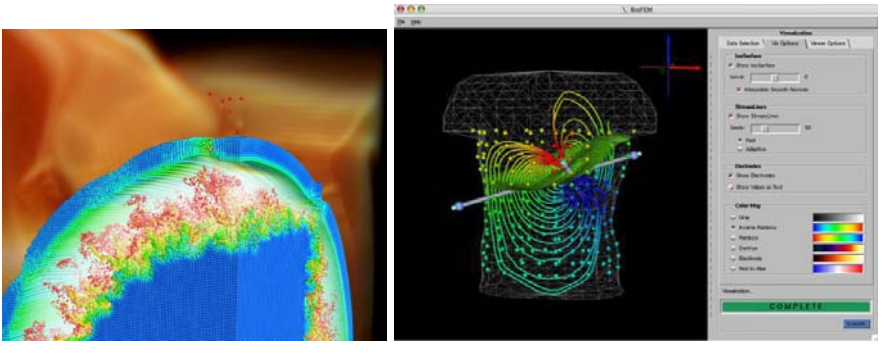


Fig. 2. C-SAFE (left): A simulation of an explosive device enveloped in a jet fuel fire, just after the point of explosion. Particles representing the solid materials (steel and HMX) are colored by temperature, and the gases (PBX product gases and fire) are volume rendered. **NCRR BioFEM PowerApp (right)**: A modeling, simulation and visualization environment for bioelectric fields. Shown here is a visualization showing the results from a finite element simulation of electric current and voltage within a model of the human torso

One of the major educational challenges posed by this environment is balancing the need to lay a firm foundation in high-performance computing "fundamentals" while at the same time exposing students to the practical issues that arise in large-scale high-performance codes as used by C-SAFE. Often times concepts and tools are taught serially across different courses and different textbooks (and with a variety of application domains in mind), and hence the interconnection between the education and the practical is not immediately apparent. Of particular importance to the mission of C-SAFE is the ability of the software to use large numbers of processors in a scalable way but also to be able to use adaptive meshes in both space and time as a means of changing resolution in order to increase the fidelity of the computation. These aims may be conflicting unless great care is taken. In Section 3.2 we present a description of a high-performance computing and parallelization course offered as part of the Computing Program which attempts to address this issue.

2.2 Center for Bioelectric Field Modeling, Simulation, and Visualization

In 2000, one of the authors (CRJ) saw the need for interdisciplinary biomedical computing research as expressed in the following [5]:

> "[R]evolutionary solutions to important science and technology problems are likely to emerge from scientists and engineers who are working at the frontiers of their respective disciplines and are also engaged in dynamic interdisciplinary interactions. . . . [B]iomedicine is now particularly well poised to contribute to advances in other disciplines and to benefit substantially from interactions with those disciplines."

In keeping with this vision, Johnson *et al.* initiated the NIH-funded Center for Bioelectric Field Modeling, Simulation, and Visualization at the University of Utah. The motivation for this Center comes from the confluence of scientific imagination and the maturation of the technology required to pursue new ideas. As computers have become more and more powerful, their users have acquired the potential ability to model, simulate, and visualize increasingly complex physical and physiological phenomena. To realize this new potential there have also been concomitant advances in computer software such as graphical user interfaces, numerical algorithms, and scientific visualization techniques. This combination of more powerful devices and the software to use them has allowed scientists to apply computing approaches to a continually growing number of important areas—such as medicine and, in particular, the important field of bioelectricity.

The mission of the Center is:

- To conduct technological research and development in advanced modeling, simulation, and visualization methods for solving bioelectric field problems.
- To create and disseminate an integrated software problem solving environment for bioelectric field problems called BioPSE [6] which allows interaction

between the modeling, computation, and visualization phases of a bioelectric field simulation as illustrated in Figure 2 (right).

One of the educational challenges within this type of environment is to develop a curriculum which instills good software engineering practices within the context of user-driven scientific computing software. Portability, extensibility, usability and efficiency all compete in this type of software environment; most Computing training focuses on one or two of these issues, but does not show how to balance the competing interests of these areas to create a product which meets the mission as stated above. The Computing degree infrastructure described in Section 3 is designed to accommodate these type of needs.

3 Integrating Research and Teaching

Students participating in high-tech research areas with our faculty are at present limited to academic program choices that do not currently reflect either the changing multidisciplinary demands of employers in industry nor the actual breadth and multidisciplinary nature of their research training and achievements. While many of these students participate in the high-quality Computer Science graduate program, their multidisciplinary needs and aspirations are somewhat different from those satisfied by conventional Computer Science, which provides more emphasis on learning about computer hardware, operating systems, and theory, and less on how to solve real-world interdisciplinary computing problems.

To bridge the gap between the high-performance programming and computing needs of the research centers as described above, we envisage an integrated research and teaching environment which provides sufficient structure to instill foundational scientific computing knowledge while providing sufficient freedom to individualize a program of study to the student's research and professional needs. The bridge has been built within the new Computing Degree offered by the School of Computing at the University of Utah, which is described in the next section.

3.1 Computing Degree Program

Two key features of our new Computing graduate degree structure are particularly designed to meet this student expectation. Not only is the new Computing degree designed to integrate knowledge from many starting points (engineering, mathematics, physics, medicine), but its track structure makes it possible to build natural and student-centered collaborative academic programs across the University. The Computing degree structure operates at both Masters and Doctoral level and is interdisciplinary through its track structure. Each track has a minimum of six faculty members who form a Track Faculty Committee. This track structure makes it possible for the Computing degree to be applicable to emerging multidisciplinary problems with a maximum of efficiency in a sound academic manner. We note that academic tracks have been shown to be a successful mechanism for offering a variety of educational opportunities within a larger degree option.

The current tracks existing under the umbrella of the Computing Degree are: (1) Scientific Computing, (2) Computer Graphics and Visualization and (3) Robotics. Our focus in this paper is on the scientific computing track.

The Scientific Computing track trains students to perform cutting edge research in all of the aspects of the scientific computing pipeline: mathematical and geometric modeling; advanced methods in simulation such as high-performance computing and parallelization; numerical algorithm development; scientific visualization; and evaluation with respect to basic science and engineering. Students apply this knowledge to real-world problems in important scientific disciplines, including combustion, mechanics, geophysics, fluid dynamics, biology, and medicine. Students integrate all aspects of computational science, yielding a new generation of computational scientists and engineers who are performing fundamental research in scientific computing, as well as being interdisciplinary "bridge-builders" that facilitate interconnections between disciplines that normally do not interact. Our mission is to provide advanced graduate training in scientific computing and to foster the synergistic combination of computer and computational sciences with domain disciplines.

The scientific computing track requires only four "fundamental" courses: Advanced Scientific Computing I/II, Scientific Visualization, and High-Performance Computing and Parallelization. These four courses are designed to provide sufficient breadth in computing issues as to allow individual faculty members to then use the remaining course hour requirements to individually direct a student's program of study to meet that student's research needs. In the following section, we describe in depth one of the four aforementioned classes, with the specific intent of showing how it fulfills the gap-filling need described earlier.

3.2 Computing Degree Program - "High-Performance Computing and Parallelization" Course

In this section we take one example from the Scientific Computing track of the new Computing degree and relate it to the C-SAFE research in high performance computing.

The course entitled "High Performance Computing and Parallelization" is intended to make it possible to understand parallel computer architecture at a high level; to write portable parallel programs using the message passing system MPI; and to understand how to construct performance models for parallel programs. The course covers the use of workstation networks as parallel computers and issues such as data decomposition, load balancing, communications and synchronization in the design of parallel programs. Both distributed memory and shared memory programming models are used. Performance models and practical performance analysis are applied to multiple case studies of parallel applications. The course is based on the books [7, 8] with background material from [9] and from a number of research papers such as [10, 4, 11, 3].

The course assignments involve writing parallel programs on a parallel computing cluster. One issue that arises in the teaching of this material is the conflict between the students being able to learn quickly and possibly interactively if at

all possible against the normal mode of batch production runs. Often the best way to resolve this conflict is through the purchase of a small teaching cluster.

Simple Performance Analysis. In understanding parallel performance it is first necessary to understand serial performance as the concepts that occur on parallel machines such as memory hierarchy are also present on serial machines in the shape of cache and *tlb* effects [8]. In understanding parallel performance and scalability the concepts of Isoefficiency, Isomemory and Isotime are all important and are often the most difficult topics for the students to grasp. Isoefficiency studies consider how fast the problem size has to grow as the number of processors grows to maintain constant efficiency. Isotime studies consider how fast the problem size has to grow as the number of processors grows to maintain constant execution time. Isomemory studies consider how fast the problem size has to grow as the number of processors grows to maintain constant memory use per processor.

These metrics may be defined for a problem of size n whose execution time is $T(n, p)$ on p processors and lead to a number of conclusions, see [10, 11]:

(i) If the Isotime function keeps (T(n,1)/p) constant then the Isotime model keeps constant efficiency, and the parallel system is scalable.
(ii) If execution time is a function of (n/p) then the Isotime and Isoefficiency functions grow linearly with the number of processors, and the parallel system is scalable.
(iii) If the Isotime function grows linearly then the Isoefficiency function grows linearly, and the parallel system is scalable.
(iv) If Isoefficiency grows linearly and the computational complexity is linear then the Isotime grows linearly, and the parallel system is scalable.

Martin and Tirado [11] quote an illuminating example from linear algebra characterized by a multigrid problem size of N^2 for which Isomemory and Isotime require $N^2 = p$ while for Isoefficiency $N^2 = p^2$. In this case if problem size is scaled with Isotime (and memory) execution time is constant and efficiency decreases slowly. In their example a 128x128 problem on 2 processors needs to go to 512x512 on 8 processors for Isoefficiency, rather than 256x256 on 8 processors for Isotime performance.

The importance of such results for C-SAFE, as it moves towards an adaptive parallel architecture of a very complex multi-physics code, is that they provides a good theoretical base for those involved in the development of the load balancing algorithms needed to make effective use of large numbers of processors on the latest generation of machines.

4 Summary and Discussion

Multidisciplinary research has become an integral part of the research landscape, and its importance will continue to grow in the future. How discipline-centered university programs adapt to the changing nature of research will directly impact

scientific and engineering progress in this next century. More tightly coupled integration of research and teaching is mandatory. The University of Utah's Computing Degree Program as described in this paper provides a mechanism solid enough to provide stability to students while progressive enough to adapt to varying needs of both the student and the research centers with which the students interact.

Acknowledgments

This work was supported by NIH NCRR grant 5P41RR012553-02 and by awards from DOE and NSF. The SCIRun and BioPSE software are available as open source from the SCI Institute website: www.sci.utah.edu.

References

[1] Jim Foley. Computing > computer science. *Computing Research News*, 14(4):6, 2002.

[2] Carleton DeTar, Aaron L. Fogelson, Chris R. Johnson, Christopher A. Sikorski, and Thanh Truong. Computational engineering and science program at the University of Utah. In *Proceedings of the International Conference on Computational Science (ICCS) 2004, M. Bubak et al, editors, Lecture Notes in Computer Science (LNCS) 3039, Part 4*, pages 1202–1209, 2004.

[3] J.D. de St. Germain, J. McCorquodale, S.G. Parker, and C.R. Johnson. Uintah: A massively parallel problem solving environment. In *Proceedings of the Ninth IEEE International Symposium on High Performance and Distributed Computing*, August 2000.

[4] S. G. Parker. A component-based architecture for parallel multi-physics PDE simulation. In *International Conference on Computational Science (ICCS2002) Workshop on PDE Software*, April 21–24 2002.

[5] Focus 2000: Exploring the Intersection of Biology, Information Technology, and Physical Systems.

[6] BioPSE: Problem Solving Environment for modeling, simulation, and visualization of bioelectric fields. Scientific Computing and Imaging Institute (SCI), http://software.sci.utah.edu/biopse.html, 2002.

[7] B. Wilkinson and M. Allen. *Parallel Programming: techniques and applications using networked workstations and parallel computers (Second Edition)*. Prentice Hall, Inc., Englewood Cliffs, N.J., 2004.

[8] S. Goedecker and M. Hoisie. *Performance Optimization of Numerically Intensive Codes*. SIAM, Philadelphia, PA, USA, 2001.

[9] P.S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, 1997.

[10] M. Llorente, F. Tirado, and L. Vázquez. Some aspects about the scalability of scientific applications on parallel computers. *Parallel Computing*, 22:1169–1195, 1996.

[11] Ignacio Martin and Fransisco Tirado. Relationships between efficiency and execution time of full multigrid methods on parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):562–573, 1997.

Education and Research Challenges in Parallel Computing

L. Ridgway Scott[1], Terry Clark[2], and Babak Bagheri[3]

[1] The Institute for Biophysical Dynamics, the Computation Institute, and the Departments of Computer Science and Mathematics, The University of Chicago, Chicago IL 60637, USA
[2] Department of Electrical Engineering and Computer Science, and Information & Telecommunication Technology Center, The University of Kansas, Lawrence, KS 66045, USA
[3] PROS Revenue Management, 3100 Main Street, Houston, TX 77002, USA

Abstract. Over three decades of parallel computing, new computational requirements and systems have steadily evolved, yet parallel software remains notably more difficult relative to its sequential counterpart, especially for fine-grained parallel applications. We discuss the role of education to address challenges posed by applications such as informatics, scientific modeling, enterprise processing, and numerical computation. We outline new curricula both in computational science and in computer science. There appear to be new directions in which graduate education in parallel computing could be directed toward fulfilling needs in science and industry.

1 Introduction

High-performance computing today essentially means parallel computing. Vector processors have a significant role to play, but even these are often grouped to form a parallel processor with vector nodes. Parallel computing has matured both as a research field and a commercial field. A look at the list of top 500 supercomputers[1] shows that there are dozens with thousands of processors. Almost all of the machines on this list (November, 2004) have over one hundred processors. These machines represent only the tip of the iceberg of parallel computers, but the size of the tip gives a hint of what lies below the surface.

The most common type of parallel computer on university campuses is a cluster of (often low cost) workstations. Many of these workstations are themselves parallel computers, with multiple processors on a single board using shared memory. At the moment, dual processor machines are the most common, but this trend may lead to larger numbers of processors available at commodity prices in a single box. Network speeds have increased (and the cost of network interface

[1] `www.top500.org`

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 44–51, 2005.
© Springer-Verlag Berlin Heidelberg 2005

cards has decreased) to the point at which a conventional network of computers in a single department or larger organizational unit can be used easily and effectively as a parallel computer for some applications. In addition, smaller clusters using a few tens of computers in a single rack with dedicated networking hardware have become the norm as computational servers for single research groups.

The use of multiple computers working on unified tasks is only increasing. Grids [1, 7] of computers extend parallel computers to a global footprint. Indeed, the grid consisting of all the computers on the planet is an available resource that is being tapped by some. The model originally pioneered by the Search for Extra-Terrestrial Intelligence (SETI), has spawned the `X@home` computational paradigm, where `X` stands for SETI (`setiathome.ssl.berkeley.edu`), folding (`folding.stanford.edu`), fightAIDS (`www.fightaidsathome.org`), predictor (`predictor.scripps.edu`), etc.

The major applications of parallel computing have broadened from the traditional area of scientific, numerically-intensive simulation. Major web servers use parallel computing to answer search queries, and computational biology servers use parallel computing to compare biological sequences. Data-intensive computation has become a major target for parallel computation. Some applications such as biological sequence analysis involve both data-intensive and computation-intensive paradigms. Even the problem of web-page ranking requires the numeric-intensive task of eigenvalue computation on a massive scale. We discuss these topics more at length in section 3.

While the applications of parallel computing continue to broaden, the core challenges of parallel computing have remained substantial and stable for at least a decade. At one point, it was hoped that compilers would solve the problem of parallel programming by automatically converting sequential programs into efficient parallel codes. However, there appears to be no magic wand in sight that will make parallel computing challenges disappear; parallel computing will remain a discipline requiring substantial study for some time.

These developments imply that education in parallel computing has an increasingly important role to play. In the past, fairly simple approaches have been appropriate. But the field has reached a level where diverse and novel ideas are needed. We provide in section 4 some suggestions for new directions to take, with an emphasis on graduate education or advanced under-graduate instruction. In a recent book [10], we have attempted to support curricula of study for parallel computing. This book could be useful in pursuing some of these ideas, but in other cases additional material would be necessary.

Parallel computing has facets which make it of interest to diverse audiences. Anyone hoping to take advantage of high-end computing today must understand parallel computing, so advanced students in any technical field requiring extensive computing will be interested to learn how to harness parallel computation for their own application areas. But parallel computing can also play a role in computer science since it involves various systems issues that can complement traditional approaches. We describe some ideas for courses that might be de-

veloped in this direction in section 4.2. Finally, parallel computing can be used simply to challenge students in new ways of thinking. Parallel computing introduces some novel mathematics as well, so it can be used to develop logical reasoning skills. We describe some mathematics that arises in parallel computing in section 4.1.

Graduate education must challenge students with open problems in order to be effective. The field is continuing to be stimulated by new problems from the scientific, engineering and commercial sectors. Graduate students must be engaged in important research questions that can help them reach the forefront of the subject. Graduate education and academic research interact synergistically when a field is still developing. Since parallel computing has been actively studied for several decades, it is reasonable to ask whether there are major research challenges left. In section 5, we describe research questions that can be posed in a beginning graduate class on parallel computing.

2 Curricular Level

There are several levels of instruction that are important. Basic instruction in computing is moving from the undergraduate to the high school curriculum, and there are efforts to move it to the middle school level [5]. Parallelism is an important concept to introduce as early as possible, but we will not address the difficult question of how early one can do this. Instead, we focus on where parallel computing might fit in new ways into existing curricula at the undergraduate and graduate levels.

Many BS and MS students will go from university to a programming job. To what extent then is it appropriate for the university to train in the craft of parallel programming? This depends on two factors. One is the market for programmers and the other is purely pedagogical. We have indicated that parallel computing has become pervasive, so the market impetus seems sufficient to justify courses on parallel computing. On the pedagogical front, one seeks courses that cause students to grow in useful ways, independent of the subject matter. Again, parallelism challenges students to confront difficult issues with both quantifiable goals and simple (if they get it right) solutions.

Minimalist treatments of parallel computing are appropriate in many cases. This type of course only requires (1) an introduction to a basic parallel programming language environment, e.g., the sequential language C and the Message Passing Interface (MPI) library for data exchange, (2) some simple parallel algorithms, and (3) an overview of parallel computer architecture. This approach is sufficient for many applications where the bulk of computation is trivially parallel. Texts are available to support a "cook book" course using popular standards. However, such an approach is not sufficient for a proper graduate course on the subject. It lacks both the intellectual depth required as well as sufficient sophistication to allow students to master difficult issues.

In moving beyond the basics, there are different directions one can take. Emphasis can be placed on algorithms, architecture, programming languages

and compilers, software engineering, and so forth. Excellent texts are available to support all of these, and faculty and students can combine different texts to achieve any desired balance. In addition, one can take an integrative approach [10] which highlights key issues in all of these areas.

3 Parallel Computing Paradigms

Diverse parallel computing paradigms make the pedagogical landscape more interesting and challenging. Much of the original impetus for parallel computing came from numeric-intensive simulation. The main issues required for success in this domain center on algorithms, programming languages and compilers, and computer architecture, especially the data-exchange network. Typically, low-latency and high bandwidth are both needed to be successful. However, data-intensive computing makes different demands.

Data-intensive computing refers to a paradigm where a large data set is a significant part of the computation. Interesting applications typically involve data sets so large that they have to be distributed across multiple processors to keep data in primary memory. Here parallelism is used to increase the memory system as much as the computational system. Software systems to support this are essential infrastructure. Often, demands on the communication system of a parallel machine are less critical than for numeric simulation. Data intensive computing may involve interaction with databases with numerous opportunities for parallelism [8]. Grid computing allows the databases in data-intensive computing to be distributed globally [4].

Data-intensive computation has been common in parts of geophysics for several decades. Companies doing data-intensive computation for oil exploration have been major consumers of parallel computers from the beginning. However, data-intensive computing is also on the rise with numerous new applications. Web servers are a simple example of data-intensive computation. However, these servers can also involve a substantial amount of numeric computation. For example, ranking the interactions of web pages, the key to a good search strategy, requires solution of an eigenvalue problem [10]. A linear system whose dimension is the number of web pages in the world (several billions are now being served) requires a good parallel solution. Biological and other sequence comparison algorithms require access to large databases as well as the usual parallel programming support required for numeric parallel computing.

Data-intensive computing can be purely integer-based, but we have shown that many applications involve a mixture of the data-intensive and the numeric-intensive paradigm. One might coin the term “data and numeric intensive computing” for this type of application. Our geophysics example falls in this class, as well as our other examples of web page ranking and biological sequence analysis.

4 New Curricula in Parallel Computing

Parallel computing introduces special challenges relative to sequential counterparts. Numerous traditional and modern areas of computer science and computational science take on new forms when parallel computing is injected as a central issue. One is rooted in parallel programming which provides a special challenge in all of these areas.

In a parallel context, topics in core computer science, including operating systems, compilers, languages, architecture, algorithms, and complexity theory, acquire characteristics unique relative to their serial form. Numerical mathematics changes focus as well when parallel algorithms become an issue; some of the issues are traditional (such as stability of new parallel variants of standard algorithms) and others are novel, since new algorithms become of interest.

Curricula can combine and focus areas to create courses in, for example, scientific and numerical algorithms, computer and network architecture, compilers and runtime systems, advanced courses in theory, and enterprise web services.

To illustrate how new curricula can be developed based on parallel computing, we give details regarding two extremes. One is mathematical and the other is about systems. One can imagine many other alternatives as well.

4.1 Math in Parallel Computing

Parallel computing introduces novel mathematical issues. These can be emphasized in a parallel computing course to challenge students in a mathematical direction. For example, a key issue in automatic parallelizing compilers is dependence analysis. This introduces some simple, but novel, issues requiring set theoretic arguments, the use of multi-index notation, and some simple number theory and algebraic geometry. It is even possible to find an application of Fermat's Last Theorem in dependence analysis [10].

The need to develop new parallel algorithms introduces mathematical challenges. However, it also offers pedagogical opportunities to introduce some advanced concepts in a simple context. For example, we develop in [10] the multigrid algorithm in one dimension. In a sequential world, this would not make any sense, since direct solution methods (e.g., Gaussian elimination) are optimal-order for one dimensional differential equation problems, and much easier to program.

Parallelizing direct methods is itself an excellent mathematical topic. It is fascinating how many different algorithms are available to handle the most important problem of sparse triangular system solution (among all the tasks in numerical linear algebra, this has the *least* inherent parallelism). Moreover, there seems to be a need for more than one algorithm to handle the full range of possible sparsity patterns [10]. Given the intrinsic difficulty of parallel direct methods, it then seems interesting to look at multi-grid as part of a parallel solution strategy. Fortunately, in one-dimension, the data structures required for multi-grid are greatly simplified and suitable for students who have not seen anything about partial differential equations.

Floating-point computation requires an understanding of stability, and establishing this for new parallel algorithms can be a source of good mathematical problems. At the introductory level, parallelization can introduce some fairly simple variants of standard algorithms, and this can lead to some useful exercises to confirm knowledge acquired in basic numerical analysis courses. At a more advanced level, there is the opportunity to be much more challenging, providing advanced numerical issues, and even research problems.

4.2 Parallel Computing as a Systems Subject

In the past, the systems curriculum in computer science was quite simple: compilers, operating systems, and databases. But now these subjects are relatively mature, and new areas are important. A key requirement of a systems course is that there be a system large and complex enough to be challenging to design and build. Parallel computing offers exactly this type of challenge.

Similarities between parallel computing and operating systems could be exploited. A curriculum could extend the classic elements typically covered throughly in operating systems (semaphores, shared-memory segments, monitors, and so on), but which are relevant also in parallel computing and programming languages. Parallel runtime systems rate more attention, and are critical to how a parallel language performs. A systems topic like this often gets limited treatment in an applications oriented course on parallel computing.

Computer science as a discipline is realizing the importance of making connections with other fields in addition to continuing to develop core areas. One way to educate students to be better prepared to do this is to present core material in a broader context. What we are suggesting here is that parallel computing provides just such an opportunity. One can imagine other areas where this could be done: a bioinformatics course that introduces key algorithms and basic learning theory, a data-mining course that covers basics in learning theory and databases, and so on.

5 Research Challenges

It is not easy to pose research questions in a beginning graduate class. However, graduate classes should provide directions that lead to research areas. In some cases, open research questions can be presented as we now illustrate.

5.1 A Long-Standing Challenge

IBM's Blue Gene project has stimulated new interest in parallel molecular dynamics [11]. Solving non-linear ordinary differential equations, such as is done in simulating planetary motion or molecular dynamics, presents a challenge to parallel computation. It has always been hard to simulate molecular dynamics on time scales that are biologically significant. Although it has been demonstrated that parallelism (using a spatial decomposition) can allow very large problems to be solved efficiently [12], the efficient use of parallelism to extend the time of

simulation for a fixed-size problem has been elusive. Decomposition of the time domain provides one possible option [2, 10].

5.2 Latency Tolerant Algorithms

The rate limiting factor in many parallel computer systems is the latency of communication. Improvements in latency have been much slower than improvements in bandwidth and computational power. Thus algorithms which are more latency tolerant are of significant interest, cf. [9].

6 Industrial Challenges

Although parallel computing has become the norm in technical computing in laboratories and academe, fine-grained parallelism is not yet widely used in commercial and corporate applications. There are many cases where fine-grained parallelism would be of great benefit. For example, pricing and revenue optimization applications process large numbers of transactions per day to generate forecasts and optimize inventory controls and prices. These computations currently need to be done in a nightly time window. Future applications would benefit from on-line analysis of trends. Whether batch or real-time, the calculations require storage of results (including intermediate results) to relational databases.

Currently, developers in industry have to use tools that are not well matched to this kind of processing. A critical requirement is standardization and broad adoption of tools. Many software developers cannot dictate the kind of hardware or operating systems that customers use. Thus a common choice for inter-process middle-ware is CORBA, which is not designed for high performance. Low-level multi-threading is often done using Java threads because of the universal acceptance of Java.

Even though MPI is a well accepted standard for technical computing, it is not yet practical in many commercial settings. The complexity of using MPI requires extensive training beyond the standard software engineering curriculum. Furthermore, debuggers and other tools would need to be available for commercial software development support. One possible improvement to the current situation might involve adoption of high-level parallel programming language constructs. Although Java provides appropriate mechanisms for a shared-memory approach, distributed-memory languages would allow the use of low-cost distributed memory machines, while still being compatible with shared-memory machines. In the past, different approaches such as High Performance Fortran [6] and the ℙ-languages [3] have been studied extensively. Adoption of the appropriate parallel constructs in popular languages might lead to a broader use of fine-grained parallelism in industry.

Graduate education can address this situation through research into more appropriate tools and systems for fine-grained parallelism. Educational programs can also transfer knowledge about existing techniques to parallel computing infrastructure and tool vendors. To be more effective in these respects in the future, parallel-computing researchers and educators may need to address the

concerns of commercial and corporate computing more directly. Some of these are (1) parallel I/O in general and parallel database access in particular, (2) standard in-memory data structures that are appropriate for both fine-grained parallel computation and database access, and (3) portability and robustness of tools.

7 Conclusions

We have indicated some directions in which graduate education could be changed in ways involving parallel computing. We have explained why this would be a good idea both for educating computationally literate people and also for solving important problems in science and industry. We outlined new curricula both in computational science and in computer science.

References

1. Abbas, A. *Grid Computing: A Practical Guide to Technology and Applications.* Charles River Media, 2004.
2. Baffico, L., Bernard, S., Maday, Y., Turinici, G., and Zérah, G. Parallel-in-time molecular-dynamics simulations. *Phys. Rev. E 66* (2002), 057701.
3. Bagheri, B., Clark, T. W., and Scott, L. R. Pfortran: a parallel dialect of Fortran. *Fortran Forum 11* (Sept. 1992), 20–31.
4. Bunn, J., and Newman, H. Data intensive grids for high energy physics. In *Grid Computing: Making the Global Infrastructure a Reality* (2003), F. Berman, G. Fox, and T. Hey, Eds., Wiley, pp. 859–906.
5. Chen, N. High school computing: The inside story. *The Computing Teacher 19*, 8 (1992), 51–52.
6. Clark, T. W., v. Hanxleden, R., and Kennedy, K. Experiences in data-parallel programming. *Scientific Programming 6* (1997), 153–158.
7. Foster, I., and Kesselman, C. *The Grid 2: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 2003.
8. Garcia-Molina, H., Labio, W. J., Wiener, J. L., and Zhuge, Y. Distributed and parallel computing issues in data warehousing. In *Proceedings of ACM Principles of Distributed Computing Conference* (1998), vol. 17, p. 7.
9. Kalé, L. V., Skeel, R., Bhandarkar, M., Brunner, R., Gursoy, A., Krawetz, N., Phillips, J., Shinozaki, A., Varadarajan, K., and Schulten, K. NAMD2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics 151* (1999), 283–312.
10. Scott, L. R., Clark, T. W., and Bagheri, B. *Scientific Parlallel Computing.* Princeton University Press, 2005.
11. Snir, M. A note on n-body computations with cutoffs. *Theory of Computing Systems 37* (2004), 295–318.
12. Wlodek, S. T., Clark, T. W., Scott, L. R., and McCammon, J. A. Molecular dynamics of acetylcholinesterase dimer complexed with tacrine. *J. Am. Chem. Soc. 119* (1997), 9513–9522.

Academic Challenges in Large-Scale Multiphysics Simulations*

Michael T. Heath and Xiangmin Jiao

Computational Science and Engineering,
University of Illinois, Urbana, IL 61801, USA
{heath, jiao}@cse.uiuc.edu

Abstract. Multiphysics simulations are increasingly playing a critical role in scientific and engineering applications. The complex and cross-disciplinary nature of such applications poses many challenges and opportunities in both research and education. In this paper we overview some of these research challenges, as well as an academic program designed to prepare students to meet them.

1 Introduction

Many physical and biological systems of interest today involve multiple interacting components with diverse space and time scales, diverse material properties, and many other sources of heterogeneity. Modeling and simulation of such systems is particularly challenging because of the diversity of knowledge and techniques required, which in turn poses a severe challenge to conventional educational programs whose compartmentalization often discourages the necessary breadth. In this paper we overview some of the research challenges arising in a fairly typical research project involving large-scale multiphysics simulations, and we also discuss an educational program designed to provide students with the cross-disciplinary expertise necessary to address such challenges successfully.

Not coincidentally, both the research project and the educational program we will describe are part of an integrated organizational framework, called Computational Science and Engineering, that has evolved at the University of Illinois as both an educational program and a research program. The goal of the educational program is to produce computationally literate scientists and engineers on the one hand, and applications-aware computer scientists on the other. Students in this educational program become "bilingual," learning the language of computing as well as the language of one or more application disciplines, such as physics, chemistry, biology, materials science, or engineering. A major goal of the research program is to enable students to experience while still on campus the kind of cross-disciplinary, team-oriented research that they are being prepared to engage in after graduation.

* Research supported by the U.S. Department of Energy through the University of California under subcontract B523819.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 52–59, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Computational Science and Engineering

At Illinois, Computational Science and Engineering (CSE) is an interdepartmental program encompassing fourteen participating departments. Students in the program receive a conventional degree (M.S. or Ph.D.) from one of these departments and also a certificate of completion of the CSE option, which is in effect analogous to a graduate minor. Core courses for the CSE option include data structures and software principles, numerical analysis, parallel programming, and scientific visualization. More advanced courses include parallel numerical algorithms, parallel computer architecture, computational mechanics, computational physics and materials science, and advanced finite element methods. Many of these courses are specifically designed to be accessible to students from multiple departments. The specific courses chosen, and the number of courses required, depend on the home department of the student, but the requirements are reasonably uniform across departments.

The courses provided by CSE play an important role in developing the breadth of expertise necessary to do true interdisciplinary research, but courses alone do not suffice. To enable students to gain first-hand experience with large-scale computation, CSE also provides computational facilities for both research and class projects. These facilities include a workstation laboratory, symmetric multiprocessor servers for shared-memory programming, and, most importantly, a very large cluster for distributed-memory programming. The current cluster owned and operated by CSE features 640 Apple Xserves—each with two G5 processors, for a total of 1280 processors—interconnected by a high-bandwidth, low-latency Myrinet network. A cluster of this size enables our students, faculty, and staff to solve very large problems and perform simulations at a scale comparable to those at the largest laboratories and supercomputer centers.

Courses can provide the relevant knowledge, and computing facilities can provide the requisite computational power, but there is still no replacement for the experience of actually *applying* them to concrete problems in science and engineering. To provide such experience, CSE also hosts multiple interdisciplinary research projects. One of these, the NSF-funded Center for Process Simulation and Design, focuses on modeling and simulation of industrial processes, such as casting and extrusion, with complex and evolving geometries. The largest project hosted by CSE is the Center for Simulation of Advanced Rockets (CSAR), funded by DOE's Advanced Simulation and Computing program, whose major goals are to advance the state of the art in computational simulation of complex systems and to train a new generation of computational scientists and engineers, which conveniently coincide with the goals of CSE. Next we will briefly describe CSAR's research program.

The principal objective of CSAR is to develop an integrated software system, called *Rocstar*, for detailed whole-system simulation of solid rocket motors [2], such as the Space Shuttle Reusable Solid Rocket Motor (RSRM) illustrated in Fig. 1. An overview of the components of the current generation of *Rocstar* is shown in Fig. 2. This system involves three broad physics disciplines—fluid dynamics, solid mechanics, and combustion—that interact with each other at the

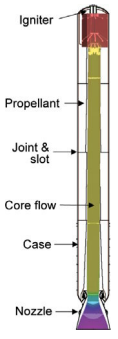


Fig. 1. Schematic of RSRM

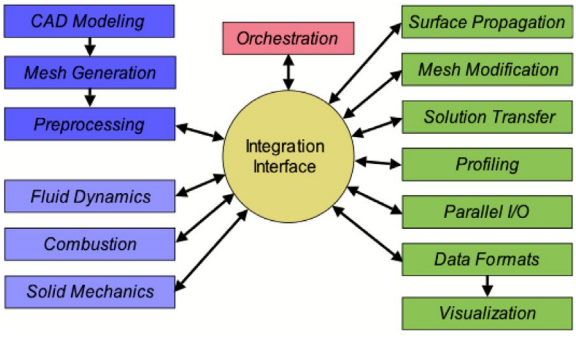


Fig. 2. Overview of *Rocstar* software components

primary system level. In addition, there are subsystem level interactions, such as particles and turbulence within fluids. The coupling issues associated with these complex interactions are representative of a large class of multiphysics simulations.

Effective coupling of multiphysics simulations poses challenges and opportunities in many areas. We will focus on two broad areas—system integration and computational mathematics—that depend crucially on two core areas of the CSE educational program, respectively software engineering and numerical analysis.

3 System Integration

Because of their complex and cross-disciplinary nature, multiphysics systems are intrinsically demanding, requiring diverse backgrounds within the research team. They are particularly challenging when the individual physics components are not merely implementations of established technologies, but are at the frontier of their respective research agendas. Moreover, component coupling schemes are challenging research subjects in their own right.

To accommodate rapidly changing and evolving systems, a software framework must allow the individual components to be developed as independently as possible, and integrate them subsequently with few or no changes. It must provide maximum flexibility for physics codes and be adapted to fit the diverse needs of the components. These requirements are at odds with many traditional software architectures and frameworks, which typically assume that the framework is fully in control, and are designed for extension instead of integration.

3.1 Component Architecture

To facilitate the diverse needs of different components, we have developed an unconventional, asymmetric architecture in which software components are grouped into the following categories:

- *Physics modules* solve physical problems in their respective geometric domains. They are similar to stand-alone applications and typically written in Fortran 90 using array based data structures encapsulated in derived types.
- *Service modules* provide specific service utilities, such as I/O, communication, and data transfer. They are typically developed by computer scientists but driven by the needs of applications, and are usually written in C++.
- *Integration interface* provides data management and function invocation mechanisms for inter-module interactions.
- *Control* (*orchestration*) *modules* specify overall coupling schemes. They contain high-level domain-specific constructs built on top of service modules, provide callback routines for physics modules to obtain boundary conditions, and mediate the initialization, execution, finalization, and I/O of physics and service modules through the integration interface.

In *Rocstar*, the above categories correspond to the components at the lower-left, right, center, and top of Fig. 2, respectively. In addition, our system uses some *off-line tools*, such as those at the upper-left corner of Fig. 2, which provide specific pre- or post-processing utilities for physics modules.

3.2 Data Management

To facilitate interactions between modules, we have developed an object-oriented, data-centric integration framework called *Roccom*. Its design is based on *persistent objects*. An object is said to be *persistent* if it lasts beyond a major coupled simulation step. In a typical physics module, especially in the high-performance regime, data objects are allocated during an initialization stage, reused for multiple iterations of calculations, and deallocated during a finalization stage. Therefore, most objects are naturally persistent in multiphysics simulations.

Based on the assumption of persistence, *Roccom* defines a registration mechanism for data objects and organizes data into distributed objects called *windows*. A window encapsulates a number of *data attributes*, such as mesh (coordinates and connectivities), and some associated field variables. A window can be partitioned into multiple *panes* to exploit parallelism or distinguish different material or boundary-condition types. In a parallel setting, a pane belongs to a single process, while a process may own any number of panes. A module constructs windows at runtime by creating attributes and registering their addresses. Different modules can communicate with each other only through windows, as illustrated in Figure 3.

The window-and-pane data abstraction of *Roccom* drastically simplifies inter-module interaction: data objects of physics modules are registered and organized into windows, so that their implementation details are hidden from the framework and need not to be altered extensively to fit the framework. Service utilities can now also be developed independently, by interacting only with window objects. Window objects are self descriptive, and in turn the interface functions can be simplified substantially, frequently reducing the number of functions or the number of arguments per function by an order of magnitude. The window

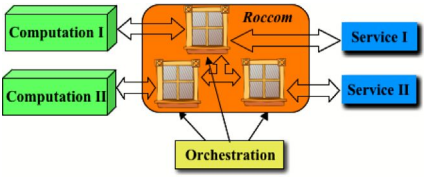


Fig. 3. Schematic of windows and panes

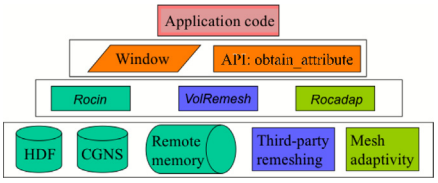


Fig. 4. Abstraction of data input

abstraction can be used for all data exchanges of a module, whether the other side is a service utility, files of various formats, or remote machines. For example, as illustrated in Fig. 4, file I/O services map *Roccom* windows with scientific file formats, and application modules obtain data from an input window through a generic function interface.

Roccom also introduces a novel concept of *partial inheritance* of windows to construct a sub-window by *using* or *cloning* a subset of the mesh or attributes of another window. In addition, the registered attributes in *Roccom* can be referenced as an aggregate, such as using "mesh" to refer to a collection of nodal coordinates and element connectivities. These advanced features allow performing complex tasks, such as reading or writing data for a whole window, with only one or two function calls. For more information on the novel features of *Roccom*, see [3].

4 Computational Mathematics

In *Rocstar*, a physical domain is decomposed into a volume mesh, which can be either block-structured or unstructured, and the numerical discretization is based on either a finite element or finite volume method. The interface between fluid and solid moves due to both chemical burning and mechanical deformation. In such a context, we must address numerous mathematical issues, three of which we discuss here.

4.1 Meshing-Related Issues

In *Rocstar*, each physics module operates on some type of mesh. A critical issue in integrated rocket simulations is the degradation of mesh quality due to the changing geometry resulting from consumption of propellant by burning, which causes the solid region to shrink and the fluid region to expand, and compresses or inflates their respective meshes. This degradation can lead to excessively small time steps when an element becomes poorly shaped, or even outright failure when an element becomes inverted. To address this issue, we take a three-tiered approach, in increasing order of aggressiveness: mesh smoothing, mesh repair, and global remeshing.

Mesh smoothing copes with gradual changes in the mesh. We provide a combination of in-house tools and integration of external packages. Our in-house effort

focuses on feature-aware surface mesh smoothing, and provides novel parallel algorithms for mixed meshes with both triangles and quadrilaterals. To smooth volume meshes, we adapted the serial MESQUITE package [1] from Sandia National Laboratories, parallelizing it by leveraging our across-pane communication abstractions.

If the mesh deforms more substantially, then mesh smoothing becomes inadequate and more aggressive mesh repair or even global remeshing may be required, although the latter is too expensive to perform very frequently. For these more drastic measures, we currently focus on tetrahedral meshes, using third-party tools off-line, including Yams and TetMesh from Simulog and MeshSim from Simmetrix. We have work in progress to integrate MeshSim into our framework for on-line use. Remeshing requires that data be mapped from the old mesh onto the new mesh, for which we have developed parallel algorithms to transfer both node- and cell-centered data accurately.

4.2 Data Transfer

In multiphysics simulations, the computational domains for each physical component are frequently meshed independently, which in turn requires geometric algorithms to correlate the surface meshes at the common interface between each pair of interacting domains to exchange boundary conditions. These surface meshes in general differ both geometrically and combinatorially, and are also partitioned differently for parallel computation. To correlate such interface meshes, we have developed novel algorithms to constructs a *common refinement* of two triangular or quadrilateral meshes modeling the same surface, that is, we derive a finer mesh whose polygons subdivide the polygons of the input surface meshes [5]. To resolve geometric mismatch, the algorithm defines a *conforming homeomorphism* and utilizes locality and duality to achieve optimal linear time complexity. Due to the nonlinear nature of the problem, our algorithm uses floating-point arithmetic, but nevertheless achieves provable robustness by identifying a set of consistency rules and an intersection principle to resolve any inconsistencies due to numerical errors.

After constructing the common refinement, data must be transferred between the nonmatching meshes in a numerically accurate and physically conservative manner. Traditional methods, including pointwise interpolation and some weighted residual methods, can achieve either accuracy or conservation, but none could achieve both simultaneously. Leveraging the common refinement, we developed more advanced formulations and optimal discretizations that minimize errors in a certain norm while achieving strict conservation, yielding significant advantages over traditional methods, especially for repeated transfers in multiphysics simulations [4].

4.3 Moving Interfaces

In *Rocstar*, the interface must be tracked as it regresses due to burning. In recent years, Eulerian methods, especially level set methods, have made significant advancements and become the dominant methods for moving interfaces [7, 8]. In

our context Lagrangian representation of the interface is crucial to describe the boundary of volume meshes of physical regions, but there was no known stable numerical methods for Lagrangian surface propagation.

To meet this challenge, we have developed a novel class of methods, called *face-offsetting methods*, based on a new *entropy-satisfying Lagrangian* (*ESL*) *formulation* [6]. Our face-offsetting methods exploit some fundamental ideas used by level set methods, together with well-established numerical techniques to provide accurate and stable entropy-satisfying solutions, without requiring Eulerian volume meshes. A fundamental difference between face-offsetting and traditional Lagrangian methods is that our methods solve the Lagrangian formulation face by face, and then reconstruct vertices by constrained minimization and curvature-aware averaging, instead of directly moving vertices along some approximate normal directions. Fig. 5 shows a sample result of the initial burn of a star grain section of a rocket motor, which exhibits rapid expansion at slots and contraction at fins. Our algorithm includes an integrated node redistribution scheme that is sufficient to control mesh quality for moderately moving interfaces without perturbing the geometry. Currently, we are coupling it with more sophisticated geometric and topological algorithms for mesh adaptivity and topological control, to provide a more complete solution for a broader range of applications.

5 Educational Impact and Future Directions

The research challenges outlined in this paper have provided substantial research opportunities for students at the undergraduate, M.S., and Ph.D. levels, and their work on these problems has enabled them to put into practice the principles and techniques learned in the courses we have described. In addition to the extensive research that has gone into designing and building the integrated simulation code, the integrated code has itself become a research tool for use by students in studying complex multicomponent physical systems. The resulting educational opportunities include the following:

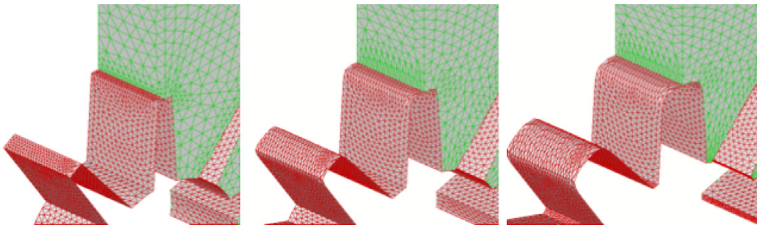


Fig. 5. Initial burn of star slice exhibits rapid expansion at slots and contraction at fins. Three subfigures correspond to 0%, 6%, and 12% burns, respectively

- Learning about interactions between different kinds of physical models using the coupled code with small problems,
- Adapting independently developed codes to include the coupling interfaces necessary to interact with other physics or service modules,
- Exploring new coupling schemes using high-level, domain-specific constructs.

Our interdisciplinary team has made substantial achievements in advancing the state-of-the-art in multiphysics simulations, but numerous challenges remain, among which are the following:

- Distributed algorithms and data structures for parallel mesh repair and adaptivity,
- Parallel mesh adaptation for crack propagation in three dimensions,
- Data transfer at sliding and adaptive interfaces.

These and other new research opportunities will require close collaboration between computer and engineering scientists to devise effective and practical methodologies. The implementation of these capabilities can be decoupled, however, owing to our flexible integration framework, and can be accomplished relatively quickly by leveraging the existing abstractions and service utilities of our infrastructure. Through this empowerment of individual students as well as interdisciplinary teams, we believe that our integration framework can play a significant role in training the next generation computational scientists and engineers.

References

1. L. Freitag, T. Leurent, P. Knupp, and D. Melander. MESQUITE design: Issues in the development of a mesh quality improvement toolkit. In *8th Intl. Conf. on Numer. Grid Gener. in Comput. Field Simu.*, pages 159–168, 2002.
2. M. T. Heath and W. A. Dick. Virtual prototyping of solid propellant rockets. *Computing in Science & Engineering*, 2:21–32, 2000.
3. X. Jiao, M. T. Campbell, and M. T. Heath. Roccom: An object-oriented, data-centric software integration framework for multiphysics simulations. In *17th Ann. ACM Int. Conf. on Supercomputing*, pages 358–368, 2003.
4. X. Jiao and M. T. Heath. Common-refinement based data transfer between non-matching meshes in multiphysics simulations. *Int. J. Numer. Meth. Engrg.*, 61:2401–2427, 2004.
5. X. Jiao and M. T. Heath. Overlaying surface meshes, part I: Algorithms. *Int. J. Comput. Geom. Appl.*, 14:379–402, 2004.
6. X. Jiao, M. T. Heath, and O. S. Lawlor. Face-offsetting methods for entropy-satisfying Lagrangian surface propagation. In preparation, 2004.
7. S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer, 2003.
8. J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

Balancing Computational Science and Computer Science Research on a Terascale Computing Facility

Calvin J. Ribbens, Srinidhi Varadarjan,
Malar Chinnusamy, and Gautam Swaminathan

Department of Computer Science, Virginia Tech,
Blacksburg, VA 24061
{ribbens, srinidhi, mchinnus, gswamina}@vt.edu

Abstract. The design and deployment of Virginia Tech's terascale computing cluster is described. The goal of this project is to demonstrate that world-class on-campus supercomputing is possible and affordable, and to explore the resulting benefits for an academic community consisting of both computational scientists and computer science researchers and students. Computer science research in high performance computing systems benefits significantly from hands-on access to this system and from close collaborations with the local computational science user community. We describe an example of this computer science research, in the area of dynamically resizable parallel applications.

1 Introduction

The importance of high-performance computing to computational science and engineering is widely recognized. Recent high-profile reports have called for greater investments in HPC and in training the next generation of computational scientists [1, 2]. Meanwhile, the raw power of the worlds fastest supercomputers continues to grow steadily, relying primarily on Moores Law and increasing processor counts. However, the appetite of computational science and engineering (CSE) researchers for high-end computing resources seems to grow faster than the ability of high performance computing (HPC) centers to meet that need. Every science and engineering discipline is making greater use of large-scale computational modeling and simulation in order to study complex systems and investigate deep scientific questions. The number of CSE practitioners is growing, and the computational demands of the simulations are growing as well, as more accurate simulations of more complex models are attempted.

Lack of access to the very high end of HPC resources is a challenge to the development of a broad CSE community. Elite researcher groups will always have access to the worlds most capable machines, and rightfully so; but identifying which groups or problems deserve that access is an inexact science. Advances often come from unexpected places. The CSE community would greatly benefit if HPC resources were available to a much broader audience, both in terms of

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 60–67, 2005.
© Springer-Verlag Berlin Heidelberg 2005

solving important problems today and in terms of training the next generation of CSE practitioners. Traditionally, the most powerful HPC resources have been located at government laboratories or federally funded supercomputer centers. Access to federal laboratory facilities generally requires collaboration with a laboratory scientist. Access to supercomputer centers is by grant processes; but these centers are often oversubscribed, and tend to favor capacity computing rather than capability computing. Meanwhile, because the most capable HPC resources are hidden behind a fence, and because the CSE practitioner community is limited in its access, computer science research in HPC systems has often been disconnected from real HPC facilities and practitioners. Although there are several notable exceptions (e.g., vectorizing compilers, computer architecture, grid computing), we believe that many areas of CS systems research have not had close day-to-day contact with the wide range of resources and practitioners in high-end CSE. None of this is surprising, of course. Historically, the most powerful HPC resources have been extremely expensive to build and operate. Hence, it makes sense that supercomputers are located where they are, and managed the way they are: they have been too precious to locate on a college campus or to replicate across many campuses; and they have been too precious to let CS systems researchers spend time on them exploring new programming models and tools, communication layers, memory systems, runtime libraries, etc.

In this paper we describe an ongoing project at Virginia Tech which seeks to address the problems described above. Our goal has been to demonstrate that world-class on-campus supercomputing is possible and affordable, and to explore the resulting benefits for our academic community—a community consisting of both computational scientists and computer science researchers and students. In Section 2 we describe the goals, design and implementation of System X. Section 3 describes one of the CS research projects that is motivated and enabled by System X. We conclude with some lessons learned in Section 4.

2 System X

Planning for Virginia Tech's Terascale Computing Facility (TCF) [3] began early in 2003. The goal was to support a rapidly growing CSE program by bringing high-end supercomputing to campus. The key challenge was affordability. While supercomputers have been invaluable to numerous science and engineering fields, their high cost—tens to hundreds of millions of dollars—has limited deployment to a few national facilities. We sought to develop novel computing architectures that reduce cost, time to build and maintenance complexity, so that institutions with relatively modest budgets could acquire their own very high-end resource.

When deployed in November 2003, the original System X consisted of 1100 Apple G5 nodes, each with two 2 GHz IBM PowerPC 970 microprocessors. That machine was ranked #3 in the 22nd Top 500 list. In October 2004 the system was upgraded, with 2.3 GHz dual Xserve G5s replacing the original nodes. This system ranks 7th on the 24th Top 500 list; it is the worlds most powerful academic machine. The system has 4.4 TB of main memory and 88 TB

of hard disk storage. The cluster nodes are interconnected with two networks: a primary InfiniBand network and a secondary Gigabit Ethernet fabric.

System X is enabling computational science and engineering researchers to tackle a wide variety of fundamental research problems, including molecular modeling, quantum chemistry, geophysics, fluid dynamics, computational biology and plasma physics. System X also provides a versatile environment for research in supercomputer systems design. Experimental cycles are set aside to enable researchers to study programming models, operating systems design, memory models, high performance networking, fault tolerance and design of distributed data storage systems.

2.1 Originality

System X was novel in several ways. First, although large-scale clusters are not new, achieving our price/performance design goals required an architecture based on untested cutting-edge technologies. None of System X's components—the Apple G5, the IBM PowerPC 970 processor, the Infiniband interconnect, the OS X operating system, and the Liebert hybrid liquid air cooling system—had ever been deployed at this scale. Furthermore, new systems software had to be written to enable the individual nodes to act in concert to form a tightly coupled supercomputer. The second novel aspect of System X is the speed at which it was constructed. Typical supercomputers of this class take eighteen months in the design and construction phases. Since our goal was to improve price/performance, we had to significantly reduce the design and build time in order to get the best performance for our limited budget. System X was designed, built and operational within three months. The third novel aspect is the cooling system. Typical supercomputing facilities and large data centers use air-conditioning technology to cool their facilities. Since the original System X consisted of 1100 nodes in a 3000 sq. ft. area, it generated a very high heat density making an air-conditioning based cooling technology very inefficient. Researchers from Liebert and Virginia Tech developed and deployed a liquid/air cooling system that uses chilled water and a refrigerant piped through overhead heat-exchangers. In this domain, the liquid air cooling technology is significantly cheaper and easier to deploy and maintain when compared to air-conditioned systems. This is the first deployment of this cooling technology.

Finally, as todays computational clusters evolve into tomorrows national cyber infrastructure, a key issue that needs to be addressed is the ability to mask component failures endemic to any large-scale computational resource. While previous generations of supercomputers engineered reliability into systems hardware, todays largest HPC platforms are based on clusters of commodity components, with no systemic solution for the reliability of the resource as a whole. For instance, if a supercomputer design is based on thousands of nodes, each of which fails only once a year, the system as a whole will fail multiple times per day. We have developed the first comprehensive solution to the problem of transparent parallel checkpointing and recovery, which enables large-scale supercomputers to mask hardware, operating system and software failures—a decades old problem.

Our system, Deja vu, supports transparent migration of subsets of a parallel application within cluster and Grid infrastructures. This enables fluid control of dynamic computational resources, where subsets of jobs transparently migrate under the control of resource aware scheduling mechanisms and distributed administrative control.

2.2 Successes and Challenges

System X achieved its design goals: price, performance, and design/construction time. The system began operations exactly three months from the Apple G5 product announcement and within three weeks of actual delivery of the nodes. The systems software stack, developed over a period of six weeks, involved researchers at Virginia Tech, Ohio State University, Israel and Japan. The completed software now provides an environment similar to other world-class supercomputers at a fraction of the cost ($5.2M), enabling researchers to port their CSE applications to System X.

The recent upgrade to the Xserve nodes has had several advantages. First, it reduces the size of the supercomputer by a factor of three, so that the machine now requires only about 1000 sq. ft. of area. Secondly, the new system consumes significantly less power than its predecessor. Third, it generates less heat, thereby reducing our cooling requirements. Fourth, the Xserve platform has automatic error correcting memory which can recover from transient bit errors. Finally, it has significant hardware monitoring capabilities—line voltages, fan speeds, communications—which allows real-time analysis of the health of the system.

Building any supercomputer presents severe logistical challenges in managing multiple aspects of the design and installation. First, funds have to be raised to finance the project. We had to articulate the research needs of the academic community and the benefits of on-campus supercomputing capabilities and make a case to administrators at Virginia Tech and the National Science Foundation to fund the project. Furthermore, with our limited track record in this area, we had to present a groundbreaking design that had the potential to succeed.

The construction of System X required detailed logistical planning and substantial infrastructure. Construction involved installing additional power and electrical equipment and cabling, bringing an additional 1.5MW of power into the facility, building and installing new cooling facilities (featuring two 125 ton water chillers), modifying the compute nodes to add communications equipment, installing communications switches and writing software to integrate the nodes into a supercomputer. In all, five equipment vendors, 80+ construction staff and 160+ student volunteers worked very hard to complete the project within three months.

Over 160 student volunteers helped in testing the nodes of the supercomputer and installing a communications card in each node. Five systems support staff helped in installing over 10 miles of high speed communications cables to interconnect the nodes. This work was completed within three weeks. Finally, we spent five weeks stabilizing and optimizing the supercomputer and writing systems software to integrate the nodes into a tightly coupled system.

3 Dynamically Resizable Parallel Applications

One of the computer science research topics we are investigating is that of dynamically resizable parallel applications. This work is directly motivated and enabled by access to an extremely large cluster such as System X. In this context, we are developing a programming model and API, data redistribution algorithms and a runtime library, and a scheduling framework. The motivation for this work stems from observations about usage of the terascale system. Given the scale of System X and the wide and unpredictable variety of jobs submitted, effective job scheduling is a challenging problem. Conventional schedulers are static, i.e., once a job is allocated a set of resources, it continues to use those resources until the end of execution. It is worth asking whether a dynamic resource manager, which has the ability to modify resources allocated to jobs at runtime, would allow more efficient resource management. In related contexts, dynamic resource management has resulted in better job and system performance (e.g., [4, 5]). Dynamic resource management enables more fine-grained control over resource usage. With dynamic resource management, resources allocated to a job can change due to internal changes in the job's resource requirements or external changes in the systems overall resource availability. In our context, dynamic resource management would extend flexibility by enabling applications to expand to a greater set of resources to take advantage of unused processors. Running applications could also shrink to a smaller subset of resources in order to accommodate higher priority jobs. The system could change the resources allocated to a job in order to meet a QoS deadline. Such a system, which enables resizing of applications, can benefit both the administrators and the users. By efficiently utilizing the resources, jobs could be completed at a faster rate, thus increasing system throughput. At the same time, by enabling applications to utilize resources beyond their initial allocation, individual job turnaround time could be improved. With this motivation in mind, the focus of our research is on dynamically reconfiguring parallel applications to use a different number of processes, i.e., on "dynamic resizing" of applications.

Additional infrastructure is required in order to enable resizing. Firstly, we need a programming model that supports resizing. This programming model needs to be simple enough so that existing code can be ported to the new system without an unreasonable re-coding burden. Secondly, runtime mechanisms to enable resizing are required. This includes support for releasing processors or acquiring new processors, and for redistributing the application's state to the new set of processors. Algorithms and a library for process and data re-mapping are described in detail in [6]. Thirdly, we require a scheduling framework that exploits resizability to increase system throughput and reduce job turn around time. The framework should support intelligent decisions in making processor allocation and reallocation in order to utilize the system effectively—by growing jobs to utilize idle processors, shrinking jobs to enable higher priority jobs to be scheduled, changing resource allocations to meet QoS deadlines, etc. In our approach the application and the scheduler work together to make resizing decisions. The application supplies preferences for the number of processors and for

processor topology; the scheduler records performance data for this application and other applications running on the system. We have extended an existing parallel scheduler [7] to interact with applications to gather performance data, use this data to make decisions about processor allocation, and adjust processor allocations to maximize system utilization. The new components in our scheduling framework include a Job Monitor, Remap Scheduler (RS), Performance Data Gatherer (PDG), and a Resize Library.

Or prototype implementation targets applications whose computation time are dominated by large ScaLAPACK [8] matrix operations. The BLACS communication layer of ScaLAPACK was modified to support dynamic process management (using MPI-2) and data and processor topology remapping. We assume that the computation is iterative, with one or more large numerical linear algebra computations dominating each iteration. Our API gives programmers a simple way to indicate "resize points" in the application, typically at the end of each iteration of an outer loop.

At resize points, the application contacts the scheduler and, if possible, provides performance data to the scheduler. Currently the metric used to measure performance is the time taken to compute each iteration. The PDG, which stores performance information for all applications currently running in the system, gathers the performance data provided by the application. This data is used to make resizing decisions. When the application contacts the scheduler, the RS makes the decision of whether to allow the application to grow to a greater number of processors, shrink the set of processors allocated to a job and reclaim the processors to schedule a different application, or permit the application to continue at its current processor allocation. A decision to shrink may be made if the application has grown to a size that has not provided a performance benefit, and hence the RS asks the application to shrink back to its previous size. An application can also be asked to shrink if there are applications waiting to be scheduled. The RS determines which running applications it needs to shrink so that an estimate of the penalty to system throughput is minimized. The RS can also allow the application to expand to a greater number of processors if there are idle processors in the system. If the RS cannot provide more processors for the application, and it determines that the application does not need to shrink, it allows the application to continue to run at its current processor allocation. The resize library, which is linked to the application, is used to perform data redistribution and construction of new processor topologies when the RS asks the application to shrink or expand. After the resize library has performed the resizing of the application, the application can perform its computation on the new set of processors. This process continues for the lifetime of the application.

We are exploring various heuristics for deciding when to resize an application, and by how much. One simple idea is to use dynamic resizing to determine a "sweet spot" for a given application. The optimal number of processors on which to run a given application is almost never known a priori. In fact, the definition of "optimal" depends on whether one cares more about throughput for a mix of jobs or turn-around time for a single job. We have implemented a job resizing

algorithm that gradually donates idle processors to a new job and measures the relative improvement in performance (measured by iteration time), in an effort to estimate the marginal benefit (or penalty) of adding (or subtracting) processors for a given application. Given this information about several long-running jobs in the system, the scheduler can than expand or contract jobs as needed, either to efficiently soak up newly available processors, or to free up under-used processors for new jobs. Simple experiments show at least a 20% improvement in overall throughput, as well as improved turn-around time for at least half of the individual jobs (none took longer) [9]. These results are highly dependent on application characteristics and job mixes of course. But the intuition behind the improvements is clear: individual jobs benefit if they would otherwise have been running on too few processors, and the entire job set benefits because the machine is utilized more effectively.

4 Conclusions

Although System X has only been in full production mode since January of 2005, we are already seeing evidence of the benefits for computer science and computational science research and graduate education. An interdisciplinary community of researchers is emerging, including both core CS investigators and applications specialists from a wide range of disciplines, including computational biology, chemistry, mechanics, materials science, applied mathematics and others. The four most important benefits are listed below, none of which would likely occur without the on-campus availability of System X:

1. **Closing the loop.** Bringing CSE practitioners and CS systems researchers together around a single high-profile resource leads to useful synergies among the two groups. CSE researchers communicate their needs to computer scientists and benefit from new systems and algorithms; CS investigators use real CSE problems and codes as motivation and test-cases for new research.
2. **Opportunities for CS research.** Research in systems and algorithms for highly scalable computing requires flexible access to a machine of such scale. Without the affordability and accessibility of Systems X, work such as that described in Section 3 would not be happening.
3. **Capability computing.** We have maintained our emphasis on capability computing, as opposed to pure capacity computing. We reserve time on the system for applications that require a substantial part of the machine for a substantial amount of time—computations that simply could not be done without this resource. The affordability and the relatively small user community, compared to national computing centers, makes this possible.
4. **Opportunities for students.** Dozens of graduate and undergraduate students are gaining valuable experience with HPC and CSE research and operations. The system provides a unique training opportunity for these students.

Despite these encouraging signs, there are still challenges facing us as we pursue the original goals of System X. In the first place, preserving the dual

mission of the facility (both computer science research and CSE applications) is not always easy, both politically and technically. Politically, we have to continue to demonstrate that experimental computer science research is as important as traditional CSE applications, both as quality research and for its benefits to future HPC applications. Technically, the challenge is to develop job management strategies and infrastructure to support both modes. For example, experimental computer science research projects may need many processors for a short period of time, while CSE applications may need a modest number of processors for a very long time. We are leveraging our fault-tolerance work to enable a sophisticated suspend/resume mechanism which allows very long-running applications to be suspended briefly, to allow experimental jobs to access the machine.

A second challenge is in finding the appropriate level of support for users. We have intentionally kept the TCF's staffing level low, in part to keep cost-recovery needs low. This means that relatively experienced HPC users get the support they need, but we do not provide extensive application-level help or help in parallelizing existing sequential codes. We do provide basic training in MPI and in using the system. Since parallelization is usually best done by someone who knows the code already, we are working to equip Virginia Tech CSE research groups to parallelize their own codes, if they so desire. This is not always an easy argument to make, however. For example, some science and engineering disciplines are slow to give credit to graduate students for writing parallel codes.

References

1. Report of the High End Computing Revitalization Task Force (HECRTF), http://www.itrd.gov/hecrtf-outreach/
2. Science and Engineering Infrastructure for the 21st Century, National Science Board, http://www.nsf.gov/nsb/documents/2003/start.htm
3. Terascale Computing Facility, Virginia Tech, http://www.tcf.vt.edu/
4. Moreira, J.E., Naik, V.K.: Dynamic Resource Management on Distributed Systems Using Reconfigurable Applications. IBM Research Report RC 20890, IBM Journal of Research and Development **41** (1997) 303–330
5. McCann, C., Vaswami, R., and Zahorjan, J.: A Dynamic Processor Allocation Policy for Multiprogrammed Shared-Memory Multiprocessors. ACM Trans. Comput. Syst. **11** (1993) 146–178
6. Chinnusamy, M.: Data and Processor Re-mapping Strategies for Dynamically Resizable Parallel Applications. MS Thesis, Virginia Tech, Dept. of Comp. Sci., 2004.
7. Tadepalli, S., Ribbens, C. J., Varadarajan, S.: GEMS: A job management system for fault tolerant grid computing. In: Meyer, J. (ed.): Proc. HPC Symposium. Soc. For Mod. and Simula. Internat., San Diego, CA (2004) 59–66
8. ScaLAPACK Project, http://www.netlib.org/scalapack/
9. Swaminathan, G.: A Scheduling Framework for Dynamically Resizable Parallel Applications. MS Thesis, Virginia Tech, Dept. of Comp. Sci., 2004.

Computational Options for Bioinformatics Research in Evolutionary Biology

Michael A. Thomas, Mitch D. Day, and Luobin Yang

Department of Biological Sciences,Idaho State University,
Pocatello, ID 83209-8007 USA
{mthomas, daymitc, yangluob}@isu.edu
http://egg.isu.edu

Abstract. This review will introduce areas of evolutionary research that require substantial computing resources using the examples of phylogenetic reconstruction and homology searching. We will discuss the commonly used analytical approaches and computational tools. We will discuss two computing environments employed by academic evolutionary researchers. We present a simple empirical demonstration of scalable cluster computing using the Apple Xserve solution for phylogenetic reconstruction and homology searching. We conclude with comments about tool development for evolutionary biology and Open Source strategies to promote scientific inquiry.

1 Introduction

An evolutionary perspective is implicit in bioinformatics approaches involving nucleotide and protein sequence analysis. For example, Dayhoff's [1] PAM matrix is based upon the assumption of different rates of amino acid substitution over evolutionary time. It is used to score sequence alignments and as a critical component of BLAST [2]. This evolutionary perspective is most developed in analyses of how families of related protein or nucleic acid sequences have diverged during evolutionary history.

The evolutionary perspective has played an indispensible role in the analysis of the human genome [3, 4] and in other genome projects (e.g., wheat, bacteria and Drosophila). Researchers identify homologs of interesting genes, infer probable gene functions by identifying conserved functional elements within genes, and determine the intensity of natural selection on a given genetic element. These approaches have also been used to explore human evolution by comparing human genes with their homologs in our relatives sharing recent ancestors (such as chimp) [5]; to investigate human disease-related genes [6]; and to identify new human disease-gene candidates through comparisons with model organisms [7].

The computational needs of molecular biologists are increasing more rapidly than our collective ability to manage, analyze, and interpret data. A number of current projects manage vocabularies describing genes and functions [8] and manage data from the numerous genome projects with generic genome database

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 68–75, 2005.
© Springer-Verlag Berlin Heidelberg 2005

construction sets [9]. The flood of freely available data means that current research requires analysis of dozens of genes and thousands of nucleotides. The added complexity of managing and analyzing this amount of data requires the skills of a bioinformatics collaborator and significant computing power. There are several distinct classes of problems that face the bioinformatics researcher. We present the examples of phylogenetic reconstruction and homology searching.

Reconstructing Evolutionary History. Phylogenetic reconstructions of the evolution of genes or species utilizes historical information contained in protein and nucleotide sequences. Phylogenetic reconstruction has made early and extensive use of bioinformatics approaches and large data sets.

There are several steps involved with phylogenetic reconstruction. First, we select an optimality criterion for selecting the proper phylogeny. Second, we examine likely trees to find those that satisfy the optimality criterion. Heuristic search algorithms are often used since an exhaustive search of all possible topologies is impossible or problematic. Third, statistical metrics determine the probability that the tree(s) found are representative of the true evolutionary relationship. See section 2 for commonly used software tools; comparisons of these tools using different cluster configurations are explored in section 3.1.

Homology Searches. Homologs are genes that share a common evolutionary ancestor. All evolutionary analyses are dependent upon the accurate and meaningful assessment of homology [10]. Large-scale genome comparisons rely on homolog predictions extracted from databases such as NCBI's HomoloGene and TIGR's EGO [5]. Predicted homologs in these databases are based on *reinforced reciprocal best match* criteria, wherein a gene from each of two organisms is shown to be the best match for the other in a genome BLAST (Basic Local Alignment Search Tool)[2].

2 Applications of Bioinformatics Tools for Phylogenetic Reconstruction

Many methods are currently used to reconstruct and evaluate phylogenetic trees [1, 11]. The nucleotides (or amino acids) are resampled with replacement to create a number of new data sets (bootstrap pseudo-replicates). An experiment may use hundreds to tens of thousands of bootstrap pseudo-replicates. For each bootstrap replicate, the tree(s) best meeting the optimality criterion are saved. For each node on the recovered tree using the original data set, the researcher records the frequency that bootstrap replicates recover the same node. Strong nodes are those that are supported by many sites in the original data set. Therefore, the bootstrap approach tests the hypothesis that the data are consistent with the tree recovered.

2.1 Optimality Criteria

The level of computational difficulty depends greatly on the type of phylogenetic analysis chosen and the criteria used to select the best tree.

Maximum Parsimony. Maximum parsimony (MP) approaches are rooted in the simple philosophical assumption that the best tree is the simplest tree that can be explained by the data. This approach ignores those nucleotide or amino acid sites that are uninformative. The researcher examines each possible (or probable) tree arrangement and calculates a tree length, which is equal to the sum of the number of changes at each site for that tree. The most parsimonious tree has the lowest tree length, but, frequently, there is more than one most parsimonious tree.

Distance Methods. Distance methods are evolutionary models intended to represent the process of evolution as it occurred on the given sequences. This method relies on a distance matrix composed of pair-wise distances for every pair of sequences in the data set using any of a number of models of nucleotide evolution. We then infer a tree from the distance matrix. Computationally, this approach is extremely efficient for most datasets.

Maximum Likelihood. Maximum Likelihood (ML) estimates are computationally intensive. Likelihood approaches calculate the likelihood that a given model is fit by a given data set. In the maximum likelihood approach to tree building, we calculate the likelihood of the data given a specific model and tree. The ML approach searches all possible tree topologies and finds the tree(s) with the highest likelihood score, a taxing task even on powerful machines.

Bayesian Analysis. A Bayesian analysis calculates the probability of the model and tree given the sequence data. We calculate this using Bayes' theorem, but the implementation is computationally intense. The Metropolis-Hastings algorithm [12, 13] can surmount this problem by using a Markov chain Monte Carlo (MCMC) search through tree space. This variation of their famous algorithm prevents the search from becoming trapped at local optima. Fortunately, this particular approach lends itself very well to parallel computing. This strategy is encoded in the *MrBayes* package (available from `http://morphbank.ebc.uu.se/mrbayes`).

2.2 Bioinformatics Tools for Phylogenetic Reconstruction

PAUP: Phylogenetic Analysis Using Parsimony (* and other methods).* PAUP is available from Sinauer (`http://paup.csit.fsu.edu/`). As the title implies, PAUP* was originally designed to implement parsimony-based approaches but has since evolved into a more comprehensive package. It is arguably the most-used and most-published bioinformatics tool for phylogenetics reconstruction. The Unix-based (command-line) portable version that can run on any Unix system, including MaxOS X. The PAUP* authors are preparing a Message Parsing Interface (MPI) designed specifically for clusters (Swofford, pers. comm.).

Phylip: Phylogeny Inference Package. Phylip is available free from (`http://evolution.genetics.washington.edu/phylip`). Phylip is a collection of dozens of open source bioinformatics applications with a unified structure and common file formats. Many developers use this open standard for structure and file formats for their own programs, building a community of users and increasing the utility of the package as a whole. Phylip has been ported to a number of platforms, but is most at home as a command-line package on a Unix machine or cluster. This characteristic makes Phylip very amenable to implementations on bioinformatics computer clusters, adding power and convenience to analyses. Phylip applications have also been integrated into BioPerl modules allowing developers to combine and extend existing tools for novel applications.

One useful tool in the Phylip package is MPI-FastDNAml, an implementation of the maximum likelihood for phylogenetic reconstruction. This program allows the user to specify a model of nucleotide evolution and searches for a phylogenetic tree that maximizes likelihood under that model. The MPI version of this program takes advantage of multiple processors to search different segments of tree space. See section 3.1 for simulation studies using this package.

Searching for Homologs. The Basic Local Alignment Search Tool (BLAST), originally created by the National Center for Biotechnology Information (NCBI), quickly approximates alignments that optimize a measure of local similarity, allowing the user to use a nucleotide or amino acid sequence to rapidly search for similar sequences in *very* large databases [2]. BLAST calculates scores of the statistical significance of alignments, providing the user a mechanism to find homologous sequences or regions. BLAST can be implemented for straightforward DNA and protein sequence database searches, motif searches, gene identification searches, and in the analysis of multiple regions of similarity in long DNA sequences. BLAST has also been extended for specialized applications.

We compare the performance of standard BLAST this program compared to three extensions and modifications (see section 3.1 for results).

3 Using Clusters for Bioinformatics

Bioinformatic problems usually have a high algorithmic complexity. ClustalW, the most commonly used multiple sequence alignment algorithm has a complexity of (n^2) where n is the number of sequences in the alignment. Consider also the number of times a project needs to be run before all the kinks and bugs are worked out. These factors combine to make the use of parallel processing power a necessity in bioinformatics research.

Beowulf for Large or Specialized Projects. A common high-performance computing solution for academic settings is a cluster of commodity-off-the-shelf (COTS) PC's yoked together in various configurations. The Beowulf approach is most commonly in academic or governmental research labs settings, but is not always ideal or cost-effective. The Beowulf approach has a low initial cost for hardware

and software, but may require a large amount of technician time for maintenance and expansion. If the effort of maintaining a Beowulf cluster itself serves an educational or research goal this effort is justifiable. However, if the primary desired product is output from high-demand computing problems the equation changes. Effort spent on care and feeding of a COTS cluster is time spent away from biological research.

Apple Xserve Clusters with iNquiry. Apple Computer sells an Apple Workgroup Cluster for Bioinformatics composed of rack-mounted Xserve G5 units with a standard, extendible platform for bioinformatics applications (iNquiry, by The Bioteam: (`http://bioteam.net`)). This system costs about $40k (education pricing) for 6 nodes including all software and hardware, rack, admin tools, etc, and requiring much less system administration experience than comparable Linux solutions. The initial hardware cost is higher, but long term costs for maintenance are lower. At our institution, this solution has proven to have the best combination of flexibility, an open, extensible platform and lower maintenance costs for our program.

3.1 Simulation Studies

The data set for these simulation studies contains 56 bacterial taxa each with 820 nucleotides from a protein-coding gene (available on request). The first study compares the performance of two programs conducting a search for the Maximum Likelihood (ML) tree, PAUP and MPI-FastDNAml. The second simulation compares four homology search programs based on the BLAST algorithm [2], NCBI-BLAST, A/G-BLAST, BT-BLAST, and MPI-BLAST. Both demonstrations were performed on a nine-node Apple Xserve computer cluster with nine nodes running the Mac OS X server platform. Five Xserve nodes were G4-based (dual 1.33Ghz with 2Gb RAM) and four were G5-based (dual 2Ghz with 4Gb RAM).

Maximum Likelihood Tree Search. MPI-fastDNAml and PAUP* searched for the Maximum Likelihood (ML) tree for our data set. Recall, the ML tree is that topology and set of branch lengths that maximizes the likelihood given the dataset and a specific model of evolution. Both programs used the simplest nucleotide substitution model that corrects for multiple substitutions (Jukes-Cantor [14]) and an heuristic search to find the ML tree.

It took PAUP* just under 40 minutes to find the tree while it took MPI-FastDNAml about 7 minutes to find the tree with 1 processor and about 3 minutes with 6 processors. FastDNAml employs an optimized heuristic search and likelihood calculation, but is less flexible in using specific models of evolution and can not take advantage of other PAUP features, such as integrating other phylogentic methods. An anticipated new version of PAUP will include refined search and likelihood algorithms and will take advantage of an MPI environment.

A/G-BLAST vs. NCBI-BLAST. We used A/G-BLAST and NCBI-BLAST to query the data set against two different large databases, NT & EST. The NT

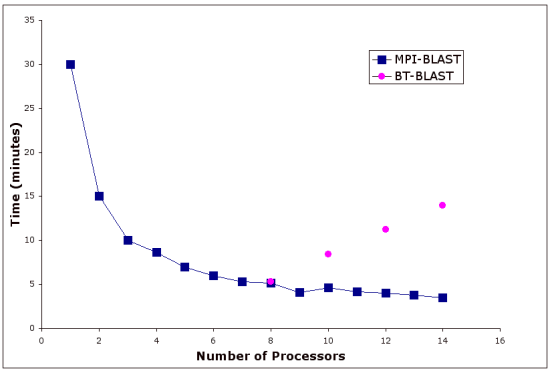


Fig. 1. MPI-BLAST vs. BT-BLAST

database includes all entries from NCBI's GenBank nucleotide database (totaling 10Gb & 2,718,617 sequences) that is nominally (but not completely) non-redundant. NCBI's EST database (8Gb & 12,582,220 sequences) includes expressed sequence tags from all species except for human and mouse, and is populated with sequences of much shorter length than the NT database. The result shows that A/G-BLAST improves the speed over NCBI-BLAST, with a much greater improvement for searches involving the small word (search target) sizes of the EST database.

Database	AG-BLAST	NCBI-BLAST
NT	26m18s	27m30s
EST	26m40s	33m37s

MPI-BLAST vs. BT-BLAST. We used our bacterial dataset to search against the NCBI NT database with different number of processors in the cluster. Both BLAST versions work in a similar fashion: the database is segmented and each processor searches one (or more) of the segments.

For MPI-BLAST, the database was divided into 14 segments. We may use any number of processors, but we limited this number to 14, equal to the number of segments in our database. MPI-BLAST does not require this limitation; we enforced this setup for comparison purposes. The number of segments and processors can be specified at run-time. Each processor will search a different segment, but some may search more segments than other processors, depending on run conditions and processors speeds. Segments are located locally - on the hard drive of the machine conducting the search of that specific segment.

For BT-BLAST, the database segmentation is more complicated (limited by size) and the number of processors used in the search *must* equal the number of segments. We divided the database into 8, 10, 12, and 14 segments. All segments are located in a single hard drive on the head node. Processors access these data via gigabit Ethernet, conduct the search, and send results to the head node.

MPI-BLAST scales very well with the number of nodes in the cluster, but BT-BLAST perforomace degrades when the nodes number increases beyond 8. There are two reasons for this. First, the first 8 processors in our cluster are G5 processors and the remainder are G4. Mixing G4 processors with G5 processors may degrade the overall performance of the cluster although this was not observed with the MPI-BLAST search. Second, BT-BLAST uses a shared directory to store database segments of. BLAST search is I/O extensive so the shared directory creates a bottleneck. This problem is specifically avoided by MPI-BLAST's distribution of the database.

4 Developing Bioinformatics Tools for Evolutionary Biology

For researchers needing to develop custom programsscripts, new software tools or even full applications, the general hardware requirements for a large cluster are essentially the same as for application-based analyses. Our language of choice for scripting and developing full-fledged bioinformatics tools is Perl. The perl interpreter will function on nearly every modern platform and properly written Perl code will run equally well on all of them. This choice is based upon the widespread use of Perl in bioinformatics research circles and specialized and general modules available through the Comprehensive Perl Archive Network (CPAN at `http://www.cpan.org`). The BioPerl modules (`http://www.bioperl.org`) available on CPAN are especially useful.

Some bioinformatics problems scale up well for parallel processing on a cluster, indeed some problems demand it. A current research effort at ISU underscores this point. We are exploring methods derived from Sequencing-by-Hybridization for their applicability to metagenomic (total environmental nucleic acids) analysis [15, 16]. This effort requires concurrent pair-wise comparisons of dozens of whole microbial genomes and oligonucleotide probe sets. The full-scale application of this method will absolutely demand the power of a cluster.

None of these advances in computing power and software development tools would be possible without the worldwide community of enthusiastic and generous contributors. Often described as the Open Source community, (`http://www.opensource.org` it represents a useful combination of generosity, pragmatism and vision that allows researchers to benefit from the collective efforts of people they may never meet. The Open Source model of software development and licensing works especially well for scientific computing in an academic research setting. The Open Source model is simply an expression of the basic ideals and values of scientific inquiry: transparency, openness and the promotion of ideas based on merit alone.

Evolutionary biology research has reached a point were access to high-powered, high-availability computing is an indispensable requirement for research. We drown in data and thirst for algorithms, tools and applications to make sense of it all. For biologists who analyze these limitless data sets, the challenge is to remain focused on the essential research tasks at hand and not be distracted

or bankrupted by the powerful computing tools at our disposal. Researchers in smaller research institutions can have access to world-class computing resources. Careful attention to the total equation of ownership and maintenance costs plus an understanding of the resources provided by the bioinformatics community and the larger world of scientific computing make it possible.

References

1. Dayhoff, M.: Survey of new data and computer methods of analysis. In Foundation, N.B.R., ed.: Atlas of Protein Sequence and Structure. Volume 5. Georgetown University, Washington, D.C. (1978)
2. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. J Mol Biol **215** (1990) 403–10
3. International Human Genome Sequencing Consortium: Finishing the euchromatic sequence of the human genome. Nature **431** (2004) 931–45
4. Imanishi, T., Itoh, T., Suzuki, Y., et al.: Integrative annotation of 21,037 human genes validated by full-length cdna clones. PLoS Biol **2** (2004) E162
5. Thomas, M., Weston, B., Joseph, M., et al.: Evolutionary dynamics of oncogenes and tumor suppressor genes: higher intensities of purifying selection than other genes. Mol Biol Evol **20** (2003) 964–8
6. Jimenez-Sanchez, G., Childs, B., Valle, D.: Human disease genes. Nature **409** (2001) 853–5
7. Feany, M., Bender, W.: A drosophila model of parkinson's disease. Nature **404** (2000) 394–8
8. GO Consortium: Creating the gene ontology resource: design and implementation. Genome Res **11** (2001) 1425–33
9. Stein, L., Mungall, C., Shu, S.Q., et al.: The generic genome browser: A building block for a model organism system database. Genome Res. **12** (2002) 1599–1610
10. Nei, M., Kumar, S.: Molecular Evolution and Phylogenetics. Oxford, New York (2000)
11. Felsenstein, J.: Inferring Phylogenies. Sinauer, New York (2004)
12. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. Journal of Chemical Physics **21** (1953) 1087–1092
13. Hastings, W.: Monte carlo s ampling methods using markov chains and their applications. Biometrika **57** (1970) 97–109
14. Jukes, T., Cantor, C.: Evolution of protein molecules. In Munro, H., ed.: Mamalian Protein Metabolism. Academic Press, New York (1969) 21–132
15. Endo, T.: Probabilistic nucleotide assembling method for sequencing by hybridization. Bioinformatics **20** (2004) 2181–8
16. Venter, J., Remmington, K., Heidelberg, J., et al.: Environmental genome shotgun sequencing of the sargasso sea. Science **304** (2004) 66–74

Financial Computations on Clusters Using Web Services

Shirish Chinchalkar, Thomas F. Coleman, and Peter Mansfield

Cornell Theory Center, Cornell University,
55 Broad Street, Third Floor, New York, NY 10004, USA
{shirish, coleman, peterm}@tc.cornell.edu

Abstract. The pricing of a portfolio of financial instruments is a common and important computational problem in financial engineering. In addition to pricing, a portfolio or risk manager may be interested in determining an effective hedging strategy, computing the value at risk, or valuing the portfolio under several different scenarios. Because of the size of many practical portfolios and the complexity of modern financial instruments the computing time to solve these problems can be several hours. We demonstrate a powerful and practical method for solving these problems on clusters using web services.

1 Introduction

The problems of financial engineering, and more generally computational finance, represent an important class of computationally intensive problems arising in industry. Many of the problems are portfolio problems. Examples include: determine the fair value of a portfolio (of financial instruments), compute an effective hedging strategy, calculate the value-at-risk, and determine an optimal rebalance of the portfolio. Because of the size of many practical portfolios, and the complexity of modern financial instruments, the computing time to solve these problems can be several hours.

Financial engineering becomes even more challenging as future 'scenarios' are considered. For example, hedge fund managers must peer into the future. How will the value of my portfolio of convertibles change going forward if interest rates climb but the underlying declines, and volatility increases? If the risk of default of a corporate bond issuer rises sharply over the next few years, how will my portfolio valuation be impacted? Can I visualize some of these dependencies and relationships evolving over the next few years? Within a range of parameter fluctuations, what is the worst case scenario?

Clearly such "what if" questions can help a fund manager decide today on portfolio adjustments and hedging possibilities. However, peering into the future can be very expensive. Even "modest" futuristic questions can result in many hours of computing time on powerful workstations. The obvious alternative to waiting hours (possibly only to discover that a parameter has been misspecified), is to move the entire portfolio system to a costly supercomputer. This

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 76–83, 2005.
© Springer-Verlag Berlin Heidelberg 2005

is a cumbersome, inefficient, and "user unfriendly" approach. However, there is good news: most of these practical problems represent loosely-coupled computations and can be solved efficiently on a cluster of processors in a master-worker framework.

We have been investigating the design of effective parallel approaches to the problems of financial engineering, and computational finance, on clusters of servers using web services. Our particular approach is to represent the portfolio in Excel with the back-end computing needs satisfied by a cluster of industry standard processors running in web services mode. The user environment we have used is Microsoft's .NET.

2 Introduction to Web Services

A web service is a piece of functionality, such as a method or a function call, exposed through a web interface ([1]). Any client on the internet can use this functionality by sending a text message encoded in XML to a server, which hosts this functionality. The server sends the response back to the client through another XML message. For example, a web service could compute the price of an option given the strike, the stock price, volatility, and interest rate. Any application over the internet could invoke this web service whenever it needs the price of such an option. There are several advantages in using web services to perform computations:

1. XML and HTTP are industry standards. So, we can write a web service in Java on Linux and invoke it from a Windows application written in C# and vice a versa.
2. Using Microsoft's .NET technology, we can invoke web services from office applications such as Microsoft Excel. This feature is especially useful in the financial industry, since a lot of end-user data is stored in Excel spreadsheets.
3. No special-purpose hardware is required for running web services. Even different types of computers in different locations can be used together as a web services cluster.
4. Since the web service resides only on the web server(s), the client software does not need to be updated every time the web service is modified. (However, if the interface changes, the client will need to be updated).
5. The web service code never leaves the server, so proprietary code can be protected.
6. Web services can be accessed from anywhere. No special purpose interface is necessary. Even a hand-held device over a wireless network and the internet can access web services.

In the context of large-scale financial computations, there are some limitations to the utilization of web services as well:

1. There is no built-in mechanism to communicate with other web services. This limits the use to loosely coupled applications.

2. The results of any computation performed during a single web service call can only be sent to the client at the end of that web service call. Thus, there is no mechanism for sending partial results while the computation is going on, without using another technology such as MPI.
3. Since messages are sent using a text format over the internet, this is not a viable computational technique for "short" computations involving a lot of data to be communicated.

Writing a web service is no different from writing a function or a method that performs the same computation. Other than a few declarative statements, there is no difference between a web service and an ordinary function. There is no reference to message passing, converting data into XML, and so on. These details are hidden from the programmer.

Similarly, invoking the web service from a client program is no different from making a function call within the same process. The relatively minor difference involves specifying in the main program the location of the web service.

3 Cluster Computing Using Web Services

A typical portfolio manager could have a large portfolio of complex instruments. These instruments may have to be priced every day. Often, several future scenarios of the stock market or interest rates may have to be simulated and the instruments may have to be priced in each scenario. Clearly, a lot of computing power is necessary. If the instruments can be priced independently of one another, we can make use of web services to perform this computation.

The entire computation can be partitioned into several tasks. Each task can consist of the pricing of a single instrument. We can have a separate web service to price each instrument. The client then simply needs to invoke the appropriate web service for each instrument. We can use other models of computation as well. For instance, in case of Monte Carlo simulation, we could split the simulations among the processors.

Figure 1 shows the overall organization of our architecture. The front-end is a typical laptop or a desktop running Excel. Data related to the portfolio is available in an Excel spreadsheet. This front-end is connected over internet or a LAN to a cluster of nodes, each of which runs a web server. When a large computation is to be performed, it is broken into smaller tasks by the Excel front-end. Each task is then shipped to an individual node which works on it independent of the other nodes. The nodes send results back to Excel, which is used to view results.

What are the advantages in using web services for cluster computing instead of MPI? There are several. First, MPI is suitable for tightly coupled problems which require significant communication between participating processes. For the financial problem described above, no inter-process communication is needed because the problem is loosely coupled. So, web services prove adequate for this task. Second, web services offer a high degree of fault tolerance. Processors can be added or taken down at any time; if a task is aborted, or if a processor is too

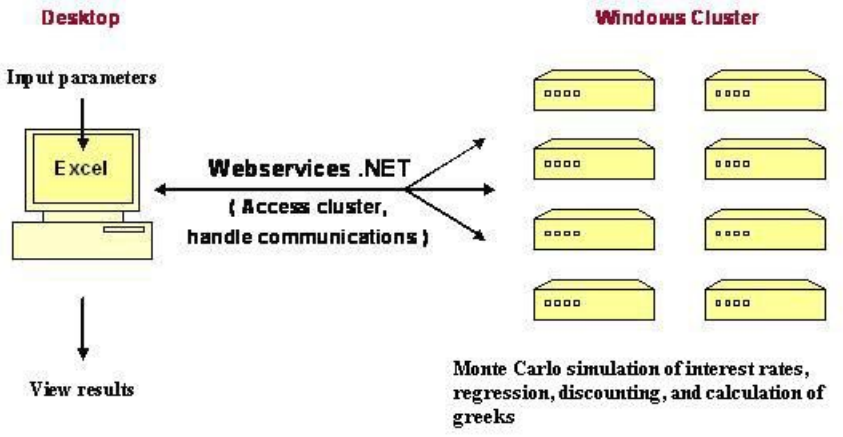


Fig. 1. Overview of computing architecture

slow, that task can be redirected to another processor. Third, in a commercial setting, the computational service (such as pricing of a financial instrument) is provided by one organization, whereas the user could be another organization or an individual. In such a setting, the user need not (and indeed should not) know the internal details of the cluster set-up, nor should he have to install any special-purpose software to use the cluster. This is possible with a web services interface.

4 Load Balancing

Given a set of tasks, we can distribute them across a .NET web services cluster in two different ways. We could send all the tasks at once to the main cluster node which uses Network Load Balancing (NLB) to distribute the tasks. However, the NLB monitors network traffic and considers those nodes that are actively communicating as busy and those that are not as idle. This is reasonable in transaction processing applications where each task can be processed quickly and the number of tasks is very large. For the problems we are interested in, we have a relatively small number of tasks, each of which takes seconds, minutes, or hours of computational time. For such problems, a node which is not sending messages might be busy doing computation and might be wrongly classified as idle by NLB. In such cases, the following approach is more suitable: We first assign tasks, one to each processor. When any task finishes, the next task is sent to the node which finished that task. This algorithm works well in practice provided there is only one application running on the cluster. If multiple applications need to run simultaneously, a centralized manager is necessary.

The load balancing mechanism described above can be implemented as a generic class, provided all data for all tasks is known before any task is executed. Fortunately, most finance applications that involve pricing portfolios of

instruments fall in this category. By using a load balancing template, we can remove from the user application, most of the low-level "plumbing" related to multi-threaded operation. This makes applications significantly easier to program and promotes code reuse.

All code related to invoking the web service asynchronously on a multi-node cluster, determining free nodes, using locks for multi-threaded operation, sending inputs, receiving results, and generating timing and speedup information is handled by the load balancing class. If the user wishes to process results as they are returned, he will need to write an application-specific callback. Again, this callback does not involve any lower-level message passing related code.

We have incorporated several useful features in the load balancing class. First, we can use multiple clusters simultaneously. These clusters can be in different locations, use different operating systems, and have different web servers. Second, if one of the nodes is down, tasks sent to that node get aborted. Once this is detected, that node is marked as down and those tasks are re-routed to one of the other nodes. Nodes marked down do not participate in further computation. Third, if, for any reason, one of the nodes is too slow, there is a provision for automatically timing out a task and re-routing it to another node. Such features are commonly required in all parallel programs and in the web services setting, they can be hidden from the application programmer very easily.

5 An Example

Cluster computing using web services as outlined above can be used to price portfolios comprising different types of instruments such as risky bonds, convertible bonds, and exotic options. We give an example which involves pricing a portfolio of callable bonds with Excel as a front-end and using multiple clusters.

A typical corporate bond has a face value, a fixed coupon, and a maturity date. Such a bond pays a fixed amount of interest semi-annually until maturity. At maturity, the face value or principal is returned[3]. A callable bond has an additional feature - the bond may be 'called back' by the issuing company by offering the bond holder or the investor an amount equal to the face value of the bond. This buy-back can be made on any of the coupon payment dates. Whether it is optimal for the issuing company to call in the bond or not depends on the prevailing interest rates and predictions of future interest rates. For example, if interest rates drop, it may be in the best interests of the issuing company to buy back the bond. If interest rates are high, the issuing company is unlikely to call in the bond. This presents two problems - first, future interest rates must be simulated, and second, the decision to buy the bond or not should be made at each coupon date, depending on the prevailing interest rate and the prediction of future interest rates.

For this work, we have used the Vasicek model for simulating interest rates. In this model, changes in interest rates are given by the formula

$$dr = a(\bar{r} - r)dt + \sigma dW \tag{1}$$

where dr is the change in the interest rate in a small time interval, dt, a is the mean reversion rate, $\bar{r}$ is the mean reversion level, and σ is the volatility. dW is a small increment of the Brownian motion, W (see [4] for more details). Given an initial interest rate, r_0, we can easily simulate future interest rates using the above equation. For valuation of callable bonds and the calculation of greeks (see below), we need several tens of thousands of simulations.

Optimal exercise along each interest rate path is determined using the Least Squares Monte Carlo algorithm, which involves the solution of a linear regression problem at each coupon date and discounting of cashflows along each interest rate path. Details of this algorithm can be found in Longstaff and Schwartz[2].

We illustrate a few additional computations for a single bond. They can be extended to a portfolio quite easily. Along with the price of the bond, we also want the bond's 'greeks'; for example bond delta and bond gamma. Delta is the first derivative of the bond price with respect to the initial interest rate ($\partial B/\partial r$) and gamma is the second derivative of the bond price with respect to the initial interest rate ($\partial^2 B/\partial r^2$), where B is the price of the bond. In this work, we have computed them using finite differences as follows

$$\Delta = \left.\frac{\partial B}{\partial r}\right|_{r=r_0} \approx \frac{B(r_0+dr) - B(r_0-dr)}{2dr} \tag{2}$$

$$\Gamma = \left.\frac{\partial^2 B}{\partial r^2}\right|_{r=r_0} \approx \frac{B(r_0+dr) - 2B(r_0) + B(r_0-dr)}{dr^2} \tag{3}$$

The above calculations require the pricing of the bond at two additional interest rates, $r_0 + dr$ and $r_0 - dr$. For all three pricing runs, we use the same set of random numbers to generate the interest rate paths (see [4]).

Once the greeks are computed, we can approximate the variation of the bond price by the following quadratic

$$B(r) \approx B(r_0) + \Delta(r - r_0) + \frac{1}{2}\Gamma(r - r_0)^2 \tag{4}$$

A risk manager would be interested in knowing how much loss this bond is likely to make, say, 1 month from now. This can be characterized by two metrics: Value at Risk (VaR) and Conditional Value at Risk (CVaR). These can be computed from the above approximation by another Monte Carlo simulation. For an introduction to VaR and CVar see [4].

The portfolio price, V, is simply a linear function of the individual bond prices

$$V = \sum_{1}^{n} w_i B_i \tag{5}$$

where the portfolio consists of n bonds, with w_i number of bonds of type i. The greeks can be computed analogously, and VaR and CVaR can be determined easily once the greeks are known.

Figure 2 shows the Excel front-end developed for this example. This interface can be used to view bond computing activity, cluster utilization and efficiency, a plot of portfolio price versus interest rate, and portfolio price, Value at Risk (VaR), Conditional Value at Risk (CVaR), and portfolio delta and gamma.

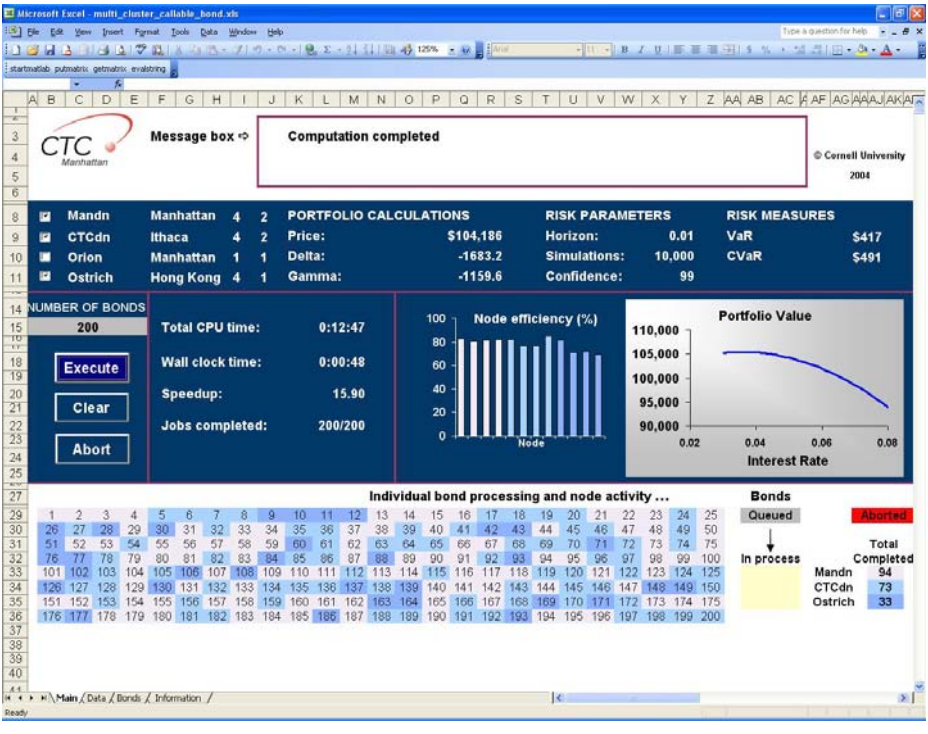


Fig. 2. Callable bond pricing on multiple clusters using Excel

In our example, the web service computes the bond price and bond greeks, whereas the Excel front-end computes the portfolio price, greeks, VaR, and CVaR. Our experiments with portfolios of as few as 200 instruments show that on 20 processors in 3 clusters at 3 different locations, we get speedups of more than 15. On a 64 processor Dell 2450 cluster consisting of 32 dual Pentium III 900 MHz processors, we have obtained speedups in excess of 60 relative to computation on a single processor for portfolios consisting of 2000 instruments, reducing 9 hours of computation on a single processor to about 9 minutes on the cluster.

6 Conclusion

Parallel computing, used to speed up a compute-intensive computation, has been under development, and in use by researchers and specialists, for over a dozen

years. Because a parallel computing environment is typically an isolated and impoverished one (not to mention very costly!), general industry has been slow to adopt parallel computing technology. Recent web services developments suggest that this situation is now improving, especially for certain application classes, such as portfolio modeling and quantitative analysis in finance. The work we have described here illustrates that a powerful analytic tool can be designed using web services technology to meet some of the computational challenges in computational finance and financial engineering.

Acknowledgements

This research was conducted using resources of the Cornell Theory Center, which is supported by Cornell University, New York State, and members of the Corporate Partnership Program. Thanks to Dave Lifka and his systems group, Yuying Li, and Cristina Patron for their help.

References

1. A. Banerjee, et. al. *C# web services - building web services with .NET remoting and ASP.NET.* Wrox Press Ltd., Birmingham, UK, 2001.
2. F. Longstaff and E. Schwartz. Valuing American Options by Simulation: A Simple Least Squares Approach. *The Review of Financial Studies*, 14:113-147, 2001.
3. Z. Bodie, A. Kane, and A.J. Markus, *Investments.* McGraw Hill. 2001.
4. P. Wilmott. *Paul Wilmott on Quantitative Finance, Volume 2.* John Wiley and Sons, New York, 2000.

"Plug-and-Play" Cluster Computing: HPC Designed for the Mainstream Scientist

Dean E. Dauger[1] and Viktor K. Decyk[2]

[1] Dauger Research, Inc., P. O. Box 3074, Huntington Beach, CA 92605 USA
http://daugerresearch.com/
[2] Department of Physics, University of California, Los Angeles, CA 90095 USA
http://exodus.physics.ucla.edu/

Abstract. At UCLA's Plasma Physics Group, to achieve accessible computational power for our research goals, we developed the tools to build numerically-intensive parallel computing clusters on the Macintosh platform. Our approach is designed to allow the user, without expertise in the operating system, to most efficiently develop and run parallel code, enabling the most effective advancement of scientific research. In this article we describe, in technical detail, the design decisions we made to accomplish these goals. We found it necessary for us to "reinvent" the cluster computer, creating a unique solution that maximizes accessibility for users. See: http://daugerresearch.com/

1 Introduction

Accessible computing power is becoming the main motivation for cluster computing. Beowulf [1], however, has taught us that the solution must be productive and cost-effective by requiring only a minimum of time and expertise to build and operate the parallel computer. Specifically, our goal is to minimize the time needed to assemble and run a working cluster. The simplicity and straightforwardness of this solution is just as important as its processing power because power provides nothing if it cannot be used effectively. This solution would provide a better total price to performance ratio and a higher commitment to the original purpose of such systems: provide the user with large amounts of *accessible* computing power.

Since 1998, we at UCLA's Plasma Physics Group have been developing and using a solution to meet those design criteria. Our solution is based on the Macintosh Operating System using PowerPC-based Macintosh (Power Mac) hardware; we call it a Mac cluster. [2] We use the Message-Passing Interface (MPI) [3], a dominant industry standard [4]. In our ongoing effort to improve the user experience, we continue to streamline the software and add numerous new features. With OS X, the latest, Unix-based version of the Mac OS, [5] we are seeing the convergence of the best of Unix with the best of the Mac.

We have extended the Macintosh's famed ease-of-use to parallel computing. In the following, we describe how a user can build an Mac cluster and demonstrate how that

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 84 – 90, 2005.
© Springer-Verlag Berlin Heidelberg 2005

user can operate it. We then describe technical details regarding important design choices we made to accomplish these design goals and the consequences of those choices, emphasizing how our solution is different from other cluster types. Part of our effort has been to rethink and streamline cluster design, installation, and operation. We believe these design principles have led us to a cluster solution that maximizes the user's accessibility to computational power.

2 The User's Experience Operating the Cluster

2.1 Building a Mac Cluster

Streamlining cluster setup to the bare minimum, the steps to building a Mac cluster have been distilled to connecting the computers to the network, assigning network names and addresses to the nodes, and quickly installing the software. The following paragraphs completely define the components and procedures for setting up a Mac cluster:

Collect the Hardware. Power Mac G4s or G5s, one Category 5 Ethernet cable with RJ-45 jacks per Mac, and an Ethernet switch. For each Mac, plug one end of a cable into the Ethernet jack on the Mac and the other end to a port on the switch.

Configure the Machines. Making sure each Mac has an working Internet or IP connection and a unique name, specified in the Network and Sharing System Preferences.

Install Software. To operate the cluster, a version of the Pooch software package is downloadable. [6] Running the installer on a hard drive of each Mac completes the parallel computer. Software installation on a node takes only a few seconds, a brevity found in no other cluster type.

2.2 Running a Mac Cluster

Because the intention is that the cluster user will spend most time interacting with the cluster performing such job launching activities, we have invested considerable effort refining the design of this user interface to minimize the time for the user to run a parallel job.

In our documentation, we recommend that users first test their Mac cluster with a simple, reliable parallel computing application such as AltiVec Fractal Carbon, available for free download. [6] This initial test also trains the user to accomplish the basic tasks required to run a parallel job. We have distilled primary cluster operation into three fundamental steps:

1. Selecting an Executable. After the user selectes New Job… from the File menu of Pooch, the user may drag the AltiVec Fractal Carbon demo from the Finder to this Job Window, depicted in Figure 1.

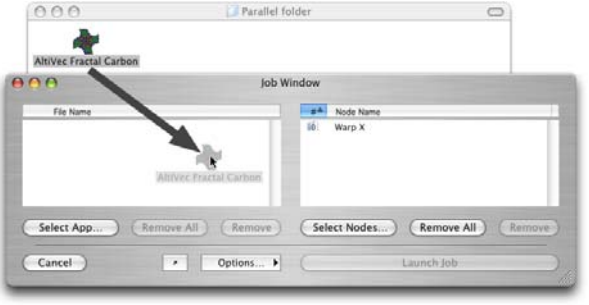


Fig. 1. To set up a parallel computing job, the user drags a parallel application, here the Fractal program, and drops it in the Job Window of Pooch

2. Selecting Computational Resources. Next, the user chooses nodes to run in parallel by clicking on Select Nodes…, which invokes a Network Scan Window, shown in Figure 2. Double-clicking on a node moves it to the node list of the Job Window.

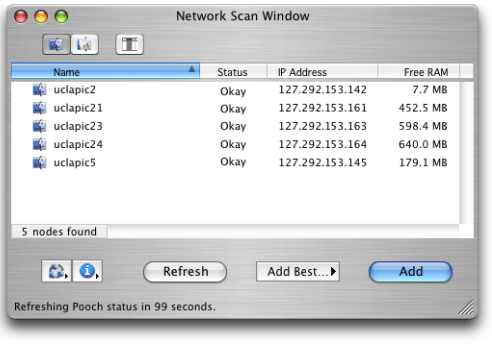


Fig. 2. Selecting nodes is performed using the Network Scan Window, invoked by clicking on "Select Nodes…" from the Job window

3. Combining These Selections Through Job Initiation. Finally, the user starts the parallel job by clicking on Launch Job.

Pooch should now be distributing copies of the parallel application to the other nodes and initiating them in parallel. Upon completion of its computational task, the demo then calculates its achieved performance, which should be significantly greater than single-node performance.

We consider the streamlining of this user interface to be important because submitting jobs is a repetitive task that potentially can occupy much of the user's time because of the intended high frequency of this task. We chose to use a graphical user interface (GUI) because a GUI tolerates the type of error and imprecision that users

can accidentally introduce when operating any device. This use of a GUI is meant to contribute to the efficiency with which the user can operate the cluster.

2.3 Debugging on a Mac Cluster

So that the Plasma group's physics researchers can maximize their time studying physics, we added enhancements, beyond basic message-passing, to the MPI implementation we call MacMPI that make it easier for them to develop parallel programs.

One of these is the monitoring of MPI messages, controlled by a monitor flag in MacMPI, which can log every message sent or received. In its default setting, a small monitor window appears, shown in Figure 3.

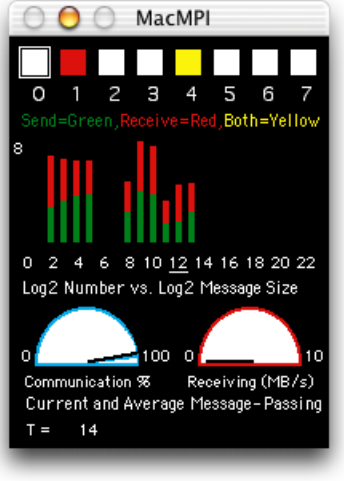


Fig. 3. The monitor window of MacMPI, which keeps track of statistics about the execution of the running parallel application

In this window, status lights indicate whether the node whose screen is being examined is sending and/or receiving messages from any other node. Green indicates sending, red indicates receiving, and yellow means both. Since messages normally are sent very fast, these lights blink rapidly. However, the moment a problem occurs, a particular color pattern is immediately visible to the user, who can then apply the new information to debugging the code.

The monitor window also shows a similarly color-coded histogram of the size of messages being sent or received. The purpose of this histogram is to draw the user's attention to the length of the messages the code is sending. The two dials in MacMPI's monitor window show the approximate percent of time spent in communication and the average and instantaneous speeds achieved during communication. While approximate, those indicators have been invaluable in revealing problems in the code and the network.

3 Design Implementation

In the design of the Mac cluster, we made the responsibilities of the communications library distinct and separate from the code that launches the jobs and manages the cluster, a separation that has existed since the Mac cluster's inception in 1998. We call the former MacMPI, while the current incarnation of the latter is called Pooch.

3.1 MacMPI

MacMPI, freely available from the AppleSeed site at UCLA Physics, is Decyk's 45 routine subset of MPI implemented using the Mac OS networking APIs. It exists in two forms: the first, MacMPI_X, uses Apple's latest Open Transport implementation of TCP/IP available in both OS 9 and OS X while the second, MacMPI_S, uses the Unix sockets implementation in OS X. [8] We achieve excellent network performance comparable to other implementations.

MacMPI is a source code library that users integrate into their executable. MacMPI is a wrapper library that assumes only the fundamental, preinstalled operating system is present and no more. MacMPI takes advantage of as much of the operating system as possible to minimize its size and complexity. We have utilized this library on hardware normally not designated for cluster operation and configured in virtually every possible configuration.

3.2 Pooch Application

Pooch is a parallel computing and cluster management tool designed to provide users accessibility to parallel computing. As of this writing, the latest version was released in September 2004. Pooch can organize the job's files into subdirectories on the other nodes and retrieve files on those nodes containing output from completed jobs. It can queue jobs and launch them only when certain conditions have been met. It also has the ability to kill running jobs, launching jobs, and queued jobs. It keeps track of these jobs and reports their status in an appealing GUI. It can also take advantage of machines where no user is logged in.

Pooch supports the widest variety of parallel programming environments, enabled by the convergence of technologies in OS X: Carbon, Cocoa, Mach-O, Unix shell scripts, and AppleScripts. [5] As of this writing, Pooch supports five different Message-Passing Interfaces (MPIs): MacMPI, mpich, MPI/Pro, mpich-gm (for Myrinet hardware), and LAM/MPI. [6] Because of OS X, MPIs of such varied histories are all now supported in the one environment.

3.3 Distinctions from Other Implementations

Division of API and Launching Utility. A fundamental difference from most other cluster types is the clear distinction and separation between the code that performs the internode communications for the job and the code that performs job initiation and other cluster management. In most MPI implementations, such as mpich and LAM, these tasks are merged in one package. Only recently has work begun on versions that identify distinctions between these tasks, such as the emerging MPICH2 rewrite of mpich. [7]

No Modification to the Operating System. Making no modifications to the operating system allowed us to simplify much of our software design. In our approach, we do not even add any runtime-linked library on the system, much less the system-level or even kernel-level modifications many cluster designs make. We took this approach so that parallel executables can run on any node regardless of such modifications. We add as little as possible to the system by adding only one additional piece of executable code, Pooch, to run and operate the cluster. This approach keeps installation time to a minimum, which helps satisfy our design goals with regards to cluster set up.

Taking Advantage of a Consistently Supported API. At UCLA Physics, we do not have the resources to build or rebuild something as complex as an operating system or the APIs it provides to applications. Therefore, we took advantage of APIs that were already present and officially supported in the Macintosh operating system. We are taking advantage of Apple's commercial, non-scientific motivation to provide a consistent, reliable, well-behaving API, operating system, and hardware.

No Static Data. No assumptions have been made about particular hardware at particular addresses being available. We rely on dynamically determined network information, automatically eliminating a host of potential sources of failure that the user might encounter. A static node list could list nodes that are in fact non-functional, and a problem is discovered only when a job fails, which could at the outset be due to a variety of potential problems in any node in the list. By making dynamic discovery part of the node selection process, problem nodes are already eliminated before the user makes a decision.

Minimum Assumptions about Configuration. The absence of further configuration details about the cluster expresses how reliably it tolerates variations in configuration while interfacing and operating with hardware and software. The configuration requirements are that the node has a working network connection with a unique IP address and a unique network name, requirements already in place for web browsing and file sharing. This design has great implications for the mainstream because end users do not wish to be concerned with configuration details.

Minimum Centralization. A common philosophy used to increase the performance of parallel codes is to eliminate bottlenecks. Extending that concept to clustering, we eliminated the "head node" of the typical Linux-based cluster. Linux clusters require shared storage (NFS, AFS, etc.) to operate, yet it is a well-known single point of failure.

We chose a decentralized approach. All nodes can act as "temporary head nodes", a transient state occurring only during the brief seconds of the launch process. If a user finds that a node is down, that user can simply move on to another node and flexibly choose how to combine nodes for cluster computation from job to job.

4 Conclusion

The inexpensive and powerful cluster of Power Mac G3s, G4s, and G5s has become a valuable addition to the UCLA Plasma Physics group. The solution at UCLA Physics is fairly unique in that half of the nodes are not dedicated for parallel computing. We

purchase high-end Macs and devote them for computation while reassigning the older, slower Macs for individual (desktop) use and data storage. Thus, we are reusing the Macs in the cluster, making for a very cost-effective solution to satisfy both our parallel computing *and* desktop computing needs. The Mac cluster is unique in this regard, made possible by how tolerant the software is of variations in configuration.

Our goal is to maximize the benefits of parallel computing for the end user. By assuming only the minimum configuration of the hardware and operating system, the Mac cluster design has the potential to provide an significant advantage to cluster users. The simplicity of using Mac cluster technology makes it a highly effective solution for all but the largest calculations. We are continuing to improve upon our work for the sake of those users and respond to their feedback.

Our approach is unique because, while other solutions seem to direct little, if any, attention to usability, tolerance to variations in configuration, and reliability outside tightly-controlled conditions, we find such issues to be as important as raw performance. We believe the ultimate vision of parallel computing is (rather than merely raw processor power) when the technology is so reliable and trivial to install, configure, and use that the user will barely be aware that computations are occurring in parallel. This article presents our progress in building the "plug-and-play" technology to make that vision come true.

Acknowledgements

Many people have provided us useful advice over the last few years. We acknowledge help given by Bedros Afeyan from Polymath Research, Inc., Ricardo Fonseca from IST, Lisbon, Portugal, Frank Tsung and John Tonge from UCLA, and the Applied Cluster Computing Group at NASA's Jet Propulsion Laboratory.

References

[1] T. L. Sterling, J. Salmon, D. J. Becker, and D. F. Savarese, How to Build a Beowulf, [MIT Press, Cambridge, MA, USA, 1999].

[2] V. K. Decyk, D. Dauger, and P. Kokelaar, "How to Build An AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing," Physica Scripta T84, 85, 2000.

[3] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, MPI: The Complete Reference [MIT Press, Cambridge, MA, 1996]; William Gropp, Ewing Lush, and Anthony Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface [MIT Press, Cambridge, MA, 1994].

[4] Most major supercomputing centers only use MPI for distributed-memory parallel computing. The absence of other message-passing schemes on new hardware is evident at NERSC: http://hpcf.nersc.gov/software/libs/ and at NPACI: http://www.npaci.edu/BlueHorizon/guide/ref.html

[5] http://www.apple.com/macosx/

[6] See http://daugerresearch.com/pooch/

[7] http://www-unix.mcs.anl.gov/mpi/mpich2/

[8] http://developer.apple.com/documentation/CoreFoundation/Networking-date.html

Building an HPC Watering Hole for Boulder Area Computational Science

E. R. Jessup[1], H. M. Tufo[2], and M. S. Woitaszek[3]

[1] University of Colorado, Boulder, CO 80309, USA
{jessup, tufo}@cs.colorado.edu
http://www.cs.colorado.edu/∼jessup
[2] University of Colorado, Boulder, CO 80309, USA
http://www.cs.colorado.edu/∼tufo
[3] University of Colorado, Boulder, CO 80309, USA
matthew.woitaszek@colorado.edu
http://hemisphere.cs.colorado.edu/∼matthew

Abstract. Access to adequate computing resources is essential for any computational science initiative. Moreover, these resources can serve as a high-performance computing watering hole where ideas are exchanged, collaborations formed, and cycles drawn upon to support interaction. We describe our efforts to bolster computational science research and education in the Boulder area by providing a computational platform for the community. We provide an overview of the educational, research, and partnership activities that are supported by, or have resulted from, this effort.

1 Introduction

At the ICCS 2004 Workshop on Computing in Science and Engineering Academic Programs, we reported on our experiences with computational science and engineering (CS&E) education at the University of Colorado at Boulder (UCB) [1]. This CS&E educational activity is part of a larger effort to bolster computational science research and education in the Boulder area with the ultimate goal of building a world class computational science institute at UCB. However, given the extremely poor financial climate at UCB, the decreasing research funding available at the national level, and the fact that computational science is still not widely embraced in academia or industry we were required to focus on low-cost activities to lay the three key pillars required for this initiative: education, external partnerships, and research.

Our initiative is supported by an inexpensive cluster-based high-performance computing platform. This platform supports educational activities, provides cycles for research (and hence future funding), and provides to our external partners a low risk environment for evaluating new ideas and equipment. In essence, this platform serves as a high-performance computing watering hole where ideas are exchanged and collaborations are formed.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 91–98, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Ten years ago acquiring such a platform would not have been feasible because of the prohibitive cost. However, the advent of clusters built using commodity off-the-shelf components (COTS) has driven down acquisition costs exponentially; it is now possible to acquire a one teraflop system with several terabytes (TB) of disk storage for approximately $250,000. Symmetrically, the advent of Linux and open source software has driven down the staffing costs to run these machines. It is our experience that such systems can be adequately administered by a handful of undergraduate and graduate students. Hence, total cost of ownership is quite small in an academic environment since grant overhead is supposed to pay for providing adequate research facilities, at least in principle.

In this paper, we describe our experiences with building a high-performance computing watering hole to support computational science in the Boulder area and discuss some of the outcomes of this effort. Specifically, we present our approaches to building the three pillars of our program. In section 2, we provide an overview of our computational facility, the means by which it was acquired, and plans to extend its capability. An overview of the educational activities supported by this facility are provided in section 3. In sections 4 and 5, we discuss the resulting partnerships and research activities, and note community involvement in section 6. Concluding remarks comprise section 7.

2 Computational Platform

Our computational platform consists of two compute clusters and a small storage cluster (see Fig. 1). The core platform was assembled over a two year period at a cost of approximately $300,000, with approximately half of that funding coming from an NSF ARI grant. It was constructed using a heterogeneous array of equipment with the goal of supporting as diverse a set of research areas as possible. For example, one cluster contains a high-performance interconnect for MPI parallel computing, another cluster uses less expensive gigabit Ethernet and is intended for batch processing of single-processor jobs, and the entire environment provides about 7 terabytes of storage for data-intensive computing.

We acquired our first cluster, Hemisphere, in January 2003. Hemisphere consists of 64 compute nodes and 2 head nodes. The compute nodes contain dual 2.4 GHz Intel P4 Xeon processors and 2 gigabytes (GB) of RAM. The compute nodes are connected using an 8 x 8 Dolphin torus for MPI communication and a gigabit Ethernet network for cluster control and I/O. We acquired our second cluster, Occam, in September 2004. It was donated by IBM via their Shared University Research (SUR) program and consists of 28 JS20 blades and a management node. Each blade contains two 1.6 GHz PowerPC 970 processors with 2.5 GB RAM and is connected by a gigabit Ethernet network for both MPI and support communication.

To make utilizing the computational platform as simple as possible, the compute clusters appear completely separate with one exception: all nodes on both clusters share the same file system space. Some users work only on Hemisphere or Occam, and some users work on both. As Hemisphere and Occam are configured

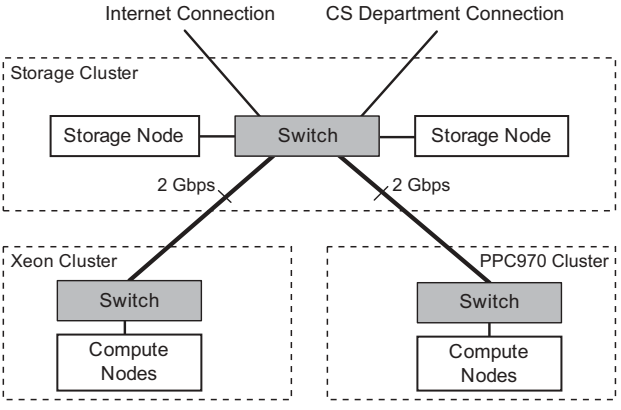


Fig. 1. UCB Computational Platform

carefully such that both systems appear similar, the shared storage allows power users to use both systems with minimal transition overhead. All users are provided with a default allocation of 1 GB in a universally accessible home directory that we backup regularly. All share an unmanaged 2.8 TB scratch workspace we call quicksand. Some users require up to 400 GB of this space, and others use none. Additional storage for large projects is available from the storage cluster provided as part of the IBM SUR grant and augmented with funds from the National Center for Atmospheric Research (NCAR). The storage cluster consists of two identical IBM x345 storage nodes, each of which provides 1.6 TB of fast Ultra320 SCSI storage. We are currently examining parallel file systems to finalize the installation with a high-performance reliable storage system.

The UCB platform is connected to NCAR via the Boulder Regional Area Network (BRAN) currently supported by OC12 fiber capable of 622 Mb/sec data transfer rates. Upgrade to gigabit Ethernet is currently underway. We use the BRAN system to connect to NCAR which will house our final computer system, CoolBlue, which is an IBM BlueGene/L (BG/L) system. This supercomputing system, to be installed in March 2005, is a one cabinet system containing 2048 compute processors arranged in an 8x16x16 3D torus, 64 I/O processors, a 4-node IBM OpenPower 720 I/O system, a p650 management node, and 8 TB of storage. At 5.7 teraflops peak (and only 12 KW) it will be our primary platform for investigating large-scale parallelism. We are currently architecting the substantial infrastructure upgrades required to seamlessly integrate CoolBlue into our computational environment.

3 HPC Educational Efforts

The Department of Computer Science at UCB was an early player in the world of CS&E education for undergraduates. Its course in high-performance scientific

computing (HPSC) was introduced in 1991. The course provided an introduction to the use of high-performance computing systems in scientific and engineering applications with undergraduate and graduate sections. Its development was supported by the National Science Foundation (NSF) under a CISE Educational Infrastructure grant awarded in 1990. The course enjoyed a successful seven year run before succumbing to technological change and a decline of student interest. The course was revived in 2002 to satisfy the increased demand for computational skills in a variety of disciplines. The source of that demand extends from academic researchers to employers in industry and the national laboratories.

To address technological developments since its first offering, we embarked on a redesign of the course. First, several structural changes were effected. To further increase the potential pool of students the numerical analysis prerequisite was removed and now appears only on the recommended course list. We moved the class project to the first semester and made HPSC a one semester course instead of two. However, as our intention was that it remain a hands-on project-based course, we kept the original four credit hour design (three hours of lecture and three hours of supervised lab per week).

In addition to structural changes, rapid changes in technology and demands from employers in industry and the national laboratories needed to be reflected in the course content and tools. Parallel programming skills are currently in high demand. As MPI is now the de facto standard for writing message-passing parallel programs we concentrate on learning MPI programming using a popular text [2]. The first half of the course is spent learning the basics of parallel programming with MPI. The remaining time is spent examining parallel architectures and algorithm development in more detail while students work on a parallel programming project of their choice. All programming and project activities revolve around using the computational platform. To encourage large-scale simulations, benchmarking, and scaling studies, dedicated access to the entire Hemisphere cluster is available through the queue. Additional HPSC course details can be found in [1].

4 Industrial and Lab Partnerships

IBM and NCAR are two Boulder area strategic partners who share an interest in promoting and strengthening computational science. IBM has been exceedingly generous in donating equipment, research support, and providing summer student internships at T. J. Watson. NCAR has supported this effort by providing money for equipment and housing equipment that is simply too costly to run at UCB. Our relationship with NCAR has also fostered the exchange of expertise and ideas in the area of computational platform management. As NCAR has over 40 years of experience running supercomputing facilities, we have learned a tremendous amount from NCAR. In return, we have used our Linux cluster expertise to help design and administer their recent cluster acquisitions.

Perhaps the most important development is the student exchange program between UCB and NCAR. Each year a select group of HPSC students are paired

with NCAR application scientists for 10-12 weeks, typically in the summer months. This offers a wonderful opportunity for cross-pollination of ideas between the scientists and newly minted high-performance computing specialists.

5 Resulting Research and Funding Initiatives

Numerous research and funding initiatives have been fostered by our high-performance computing watering hole. We concentrate on those which have resulted in significant funding and are illustrative of our research interests in large-scale parallel computing, software tools to support HPC systems, and scientific application development. The following are brief overviews of four such research initiatives.

5.1 IBM BlueGene/L

In December 1999, IBM Research launched a multi-year and multi-disciplinary project called BlueGene. It is an ambitious project that currently involves more than 50 IBM researchers in laboratories worldwide. One of the stated goals of this project is to investigate biologically important phenomena such as protein folding. An equally important goal is to develop the next generation of Petascale high-performance computing architectures. In November 2001, IBM announced a partnership with Lawrence Livermore National Laboratory (LLNL) to build the BlueGene/L supercomputer, a new architecture for high performance parallel computing systems based on low cost, low power embedded PowerPC technology. The LLNL BlueGene/L system will have 65,536-nodes each capable of 5.6 Gigaflops peak performance. BlueGene/L has several promising characteristics relative to current Terascale systems. First, BlueGene/L's overall cost-performance ratio is expected to be about an order of magnitude less than the Earth Simulator's. Though it will appear three years after the Earth Simulator, its peak floating point rate is expected to be about 9 times higher, representing more than a factor of two improvement over what Moore's Law would predict. BlueGene/L has a very fast combining network that will be useful for broadcast and reduction operations, which are a weak point of all current large-scale clusters.

In collaboration with NCAR and the University of Colorado at Denver, we submitted a proposal to the NSF Major Research Instrumentation program to acquire a one cabinet BlueGene/L supercomputer. This collaboration involved twelve researchers at the three institutions. The primary goal of the project is to investigate and address the technical obstacles to achieving practical Petascale computing in geoscience, aerospace engineering, and mathematical applications using the IBM BlueGene/L system as the target compute platform. Specific topics include: scalable high-order methods for climate modeling, dynamic data driven wildfire modeling, high-fidelity multidisciplinary simulations of modern aircraft, and scalable domain-decomposition and multigrid solvers.

5.2 Grid

Another successful collaboration has been the creation of a prototype Grid between UCB and NCAR using the Globus Toolkit [3]. This grid supports work on the Grid-BGC project [4], which is funded by a three year NASA Advanced Information Systems Technology (AIST) grant. The objective of the Grid-BGC project is to create a cost effective end-to-end solution for terrestrial ecosystem modeling. Grid-BGC allows scientists to easily configure and run high-resolution terrestrial carbon cycle simulations without having to worry about the individual components of the simulation or the underlying computational and data storage systems. In the past, these simulations needed to be performed at NCAR because a direct connection to the 2 petabyte mass storage system was required. With the introduction of Grid technologies, this is no longer the case, and the simulations may be performed on less expensive commodity cluster systems instead. In addition to outsourcing compute jobs, Grid projects allow our students and facilities to contribute to application science.

In contributing to the Grid-BGC project we discovered that Grid-related projects are becoming increasingly popular in the academic and research community. As a class project for the previous year's HPSC course, a group of students associated with NCAR's Mesoscale Microscale Meteorology (MMM) division (outside the purview of our existing collaborative relationship) worked on grid-enabling a portion of a meteorology model for simultaneous execution on Hemisphere and a NCAR cluster operated by the MMM group. Our internal experiences with Grid computing have proven to be helpful in providing expertise, on top of our computational resources, to other research groups at UCB and NCAR.

5.3 Shared Parallel File Systems

We are currently involved in a collaborative research project with NCAR to construct and evaluate a centralized storage cluster using current commercially available parallel file system technology [5]. Traditional single-host file systems (e.g., those exported via Network File System (NFS)) are unable to efficiently scale to support hundreds of nodes or utilize multiple servers. We are performing a detailed analysis of IBM's General Parallel File System (GPFS), Cluster File Systems' (CFS) Lustre, TerraScale Technologies' TerraFS, and Argonne and Clemson's Parallel Virtual File System 2 (PVFS2) for use in an environment with multiple Linux clusters running with different hardware architectures and operating system variants.

A single shared file system, with sufficient capacity and performance to store data between use on different computers for processing and visualization, while still meeting reliability requirements, would substantially reduce the time, network bandwidth, and storage space consumed by routine bulk data replication. This would provide a more user-friendly computing environment, allowing scientists to focus on the science instead of data movement. While this study is of interest to NCAR, the experimentation will help us finalize the configuration of our own storage cluster. We are also proposing to construct a prototype distributed machine room to address issues to storage and computing in a geograph-

ically separated environment. Such capabilities will be required for integrating CoolBlue into our computational facility.

5.4 Scalable Dynamical Cores

We have been working with IBM and NCAR researchers to build a scalable and efficient atmospheric dynamical core using NCAR's High Order Method Modeling Environment (HOMME) [6]. In order for this to be a useful tool for atmospheric scientists it is necessary to couple this core to physics packages employed by the community.

The physics of cloud formation is generally simulated rather crudely using phenomenological parameterizations. The dream of modelers is the direct numerical simulation of cloud processes on a global scale. Unfortunately, this requires an increase in computational power of approximately six orders of magnitude over what is currently available. A promising alternative to improve the simulation of cloud processes in climate models is a compromise technique called Cloud Resolving Convective Parameterization (CRCP, also known as Super-Parameterization). The cost of this approach is two to three orders of magnitude more expensive than traditional parameterization techniques. However, with the advent of BlueGene/L this is now tractable. NCAR researchers have built a super-parameterization package and work is underway to couple this to HOMME. The goal is to produce an atmospheric model capable of exploiting BG/L's scalability and computational power to realize practical and scientifically useful integration rates for super-parameterized climate simulation.

With help from research staff at the IBM Watson Research Center and at NCAR we have ported HOMME and CRCP to a four rack BlueGene/L system at Watson, scaled the code to approximately eight thousand processors, and achieved sustained performance of approximately 15% of peak. Through our partnership with IBM we will have access to half of the LLNL system as it is being built this spring in Rochester, Minnesota.

6 Community Research Activities

While our computational platform has provided our research group multiple opportunities for partnerships and funded research initiatives, we are actually the smallest consumer of our computational resources. We actively seek and encourage other research groups to make use of our equipment. Through exposure via the HPSC course and other channels, several groups on campus have discovered and utilize our computational platform.

The Computer Science department's numerical analysis group runs large parallel jobs for testing a new approach to Lagrange-Newton-Krylov-Schwarz class of parallel numerical algorithms, and the systems research group runs thousands of single processor jobs to simulate the behavior of wireless networks. The Electrical and Computer Engineering department similarly runs parameter studies to simulate and analyze experimental microprocessor run-time activity. The CU Center for Aerospace Structures uses the computational platform to examine

aeroelastic optimization methodologies for viscous and turbulent flows, the Department of Molecular Cellular and Developmental Biology uses the clusters to locate cellular structures present in three-dimensional digital images generated by electron microscopes, and the Solid Earth Geophysics research group uses the clusters to develop codes for cross correlation of global passive seismic data for tomography and geophysical inversion problems. In this sense, our computational platform has evolved from a single group's private cluster to a community resource. We provide managed, reliable, computing facilities at no cost, and the community response has been encouraging.

7 Conclusions

Though this effort is only two years old we have made great strides to provide a substantial set of computational resources to the Boulder Area scientific computing community to serve as a high-performance computing watering hole. This facility is at the core of our HPC educational activities and has initiated and supported a diverse set of research projects with investigators at UCB and at our strategic partners NCAR and IBM.

Acknowledgments

Support for the the University of Colorado Computational Science Center was provided by NSF ARI Grant #CDA-9601817, NSF MRI Grant #CNS-0420873, NASA AIST grant #NAG2-1646, DOE SciDAC grant #DE-FG02-04ER63870, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research (SUR) program. We would like to especially thank NCAR and IBM for their continued support, and in particular, Rich Loft (NCAR), Al Kellie (NCAR), Christine Kretz (IBM) and Kirk Jordan (IBM).

References

1. Jessup, E.R., Tufo, H.M.: Creating a Sustainable High-Performance Scientific Computing Course. In: International Conference on Computational Science. (2004) 1242
2. Pacheco, P.: Parallel Programming with MPI. Morgan Kaufmann, San Francisco, CA (1997)
3. Alliance, T.G.: The Globus Project (2003) http://www.globus.org/.
4. Cope, J., Hartsough, C., Tufo, H.M., Wilhelmi, N., Woitaszek, M.: GridBGC: A Grid-Enabled Terrestrial Carbon Cycle Modeling System. In: Submitted to CCGrid - IEEE International Symposium on Cluster Computing and the Grid. (2005)
5. Cope, J., Oberg, M., Tufo, H.M., Woitaszek, M.: Shared Parallel Filesystems in Heterogeneous Linux Multi-Cluster Environments. In: Submitted to 6th LCI International Conference on Linux Clusters: The HPC Revolution. (2005)
6. St-Cyr, A., Dennis, J.M., Tufo, H.M., Thomas, S.J.: An Efficient Non-Conforming Spectral Element Atmospheric Model using Nonlinear Operator Integration Factor Splitting. (2004)

The Dartmouth Green Grid

James E. Dobson[1,⋆], Jeffrey B. Woodward[1], Susan A. Schwarz[3], John C. Marchesini[2], Hany Farid[2], and Sean W. Smith[2]

[1] Department of Psychological and Brain Sciences, Dartmouth College
[2] Department of Computer Science, Dartmouth College
[3] Research Computing, Dartmouth College

Abstract. The Green Grid is an ambitious project to create a shared high performance computing infrastructure for science and engineering at Dartmouth College. The Green Grid was created with the support of the Dean of the Faculty of Arts & Sciences to promote collaborative computing for the entire Dartmouth community. We will share our design for building campus grids and experiences in Grid-enabling applications from several academic departments.

1 Introduction

A Campus Grid enables the collaboration of multiple departments, labs, and centers within an institution. These groups are often in their own administrative domains but can share some common infrastructure. Dartmouth College has built a campus grid architecture called the "Green Grid" to leverage this shared infrastructure. Campus Grids can often be constructed faster than a larger multi-institutional Grid since there are common infrastructure services such as high speed networking, naming services (DNS), and certificate authorities already established. Dartmouth was an early adopter of Public Key Infrastructure (PKI) technology which is leveraged by the Globus software stack. The Green Grid would not have been possible without a close collaborative relationship between the departments providing these services including Network Services, the Dartmouth PKI Lab, and Research Computing. There are several existing campus grid projects at Virginia Tech [1], University of Michigan [2], University of Wisconsin, and University of Buffalo. In building the Green Grid we have attempted to follow the conventions established by the Global Grid Forum [3], the National Science Foundation's Middleware Initiative (NMI), several large-scale national Grid projects [4, 5] and the work already done by existing campus grid projects.

2 Campus Grid Design

The Green Grid was designed to be a simple architecture which could expand to include the many scientific Linux-based clusters distributed around the Dartmouth campus. We followed two major design principles for the Green Grid:

⋆ Contact Author: HB 6162, Hanover, NH 03755 James.E.Dobson@Dartmouth.EDU

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 99–106, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Table 1. Dartmouth's Campus-wide Grid Resources (Estimated)

Department	CPUs Phase I	CPUs Phase II	CPUs Phase III
Math	12	20	100
Research Computing	12	32	128
Tuck School of Business	12	12	128
Biology	12	12	128
Psychological and Brain Sciences	12	32	128
Computer Science	12	64	512
Physics	12	50	128
ISTS	12	16	128
Chemistry	12	12	60
Dartmouth Medical School	12	500	600
Total	60	750	1912

No Major Centralized Infrastructure. The Green Grid must exist as an informal collaboration between departments. No major systems or centralized infrastructure should be needed for the Grid to be operational. The groups providing resources should be able use their systems disconnected from the Green Grid. We did not want to create new authentication systems or stray too far from the standard technologies already deployed at the campus level.

Local Control over Local Resources. For the Green Grid to successfully integrate the clusters owned by individual departments and PI's the architecture needed to enable the local cluster administrators to keep control over their own resources. The Green Grid doesn't require specific schedulers or resource management systems to be used. System administrators need to implement a small stack of software but not replace existing resource management systems.

2.1 Project Phases

Phase I. The initial project phase included the purchase of 60 dual processor systems to bootstrap a Grid computing infrastructure. This system served as a reference architecture and immediately provided 120 processors (Table 1) available for running applications.

Phase II. The second phase of the Green Grid project extends the Grid to include the dedicated Linux clusters housed within each department. The several labs of Linux desktops will be added to this infrastructure. Departments beyond the initial 10 are coming online with their own computer resources. We are currently in this phase of the project. The application requirements for the new users are being taken into effect and the final software stack is being defined for the cluster owners and system administrators to deploy.

Phase III. We plan to look for solutions for extending the Green Grid beyond dedicated servers and clusters. The thousands of desktops on the campus could be integrated with

the Green Grid infrastructure for the running of batch applications. In the previous project phases we assume that the execution hosts will be running a Linux distribution on an x86 or x86 compatible system. In Phase III we will have to deal with true heterogeneity.

2.2 Certificate Management

The Dartmouth PKI Lab, initially chartered by Internet2, has been pursuing both development and deployment of campus PKI as well as experimental research. Both aspects of this work are being integrated into the Green Grid.

On the deployment end, we've done a phased roll-out of X.509 identity certificates to the campus population, and retrofitted principal campus information services to use PKI-based authentication and authorization; for some sensitive administrative services, PKI is the only option now permitted. Initially, Dartmouth Certification Authority (CA) issued low-assurance certificates online, by confirming the identity of the remote user via the college's legacy campus-wide userid/password system. Recently, the Dartmouth CA started offering high-assurance certificates, required for higher-assurance services. In addition to the standard identity lookup, the CA requires a form of in-person authentication before it will issue a high-assurance certificate; in some versions, the user's private key is protected inside a USB dongle.

The Green Grid bootstraps on this PKI. When a new client installation is done, the user can have the software enroll him or her in the Dartmouth PKI: we obtain the user's username and password, and post these, over HTTPS, to the CA's Web-based enrollment system, and receive a low-assurance certificate. If the client already has a Dartmouth-certified keypair and the keystore permits export, the client can opt to export and use that keypair instead of getting a fresh one. Our current plan is to offer a MyProxy[6] service for users to store their Green Grid credentials.

The Green grid is also bootstrapping on the PKI Lab research work. For example, our *Secure Hardware Enhanced MyProxy (SHEMP)* [7] project takes advantage of TCPA/TCG secure hardware and the eXtensible Access Control Markup Language (XACML) [8] to harden a MyProxy credential repository. This system allows a user to specify the conditions under which the repository should use her private key, and for what purposes. We plan on piloting this within the Green Grid.

3 Applications

The Green Grid is an important research tool which is bringing computational scientists, students, and researchers across the campus together on a shared computer platform.

- Bioinformatics
  A number of research groups at Dartmouth are using gene sequencing. One researcher plans to sequence $>10{,}000$ random Damselfly genes to construct a list of genes that may be involved in genetic networks mediating the responses. This application will access data stored in a network database. Another Bioinformatics application takes a sample dataset and randomizes a column $1{,}000$ times to create new input datasets which are then run through the Green Grid.

– Math
  One of the first applications to be run on the Green Grid was C code from the Math department which was graduate student project to search for special Proth primes with $1,500$ digits. This code was compiled for the x86-64 architecture using the GNU C compiler and the GNU GMP library. This application was a single static binary which was able to read data and write from standard UNIX facilities.
– Virtual Data System
  The Virtual Data System [9] (VDS) from the IVDGL and GriPhyN [10] projects is used, in part, for some of the applications running on the Green Grid. Virtual Data is being integrated into applications and methods used in the research labs within Psychological and Brain Sciences. Site selection for VDS is done using a random selection from an array of sites (departmental Globus Gatekeeper nodes) which can run the requested application. We have run a spatial normalization workflow (Fig 1) on four sites during the initial test runs.

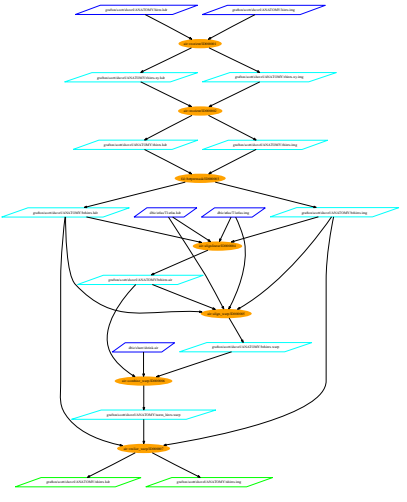


Fig. 1. Example fMRI Workflow

3.1 Software

The Globus Gatekeeper nodes on the Green Grid are all using the Globus Toolkit version 3.2.0. We have made a few simple modifications to both the Gatekeeper and the GridFTP server to obtain multiple AFS tokens. The Green Grid uses pre-WS Globus services such as GRAM. We are using MDS-2 for a Grid information service. Each departmental Globus Gatekeeper node that is also reporting into a GRIS server. web-interface (mdsweb) is used to display data from each of the departmental Gatekeepers. In addition to the standard Globus services we are also using the the GSI-SSH package for remote shell access and small file transfers. Our current distribution of the grid-mapfile is through a http server. In the future we would like to use a relational database to store our authorization data using both the SAZ[11] and VOMS services.

3.2 Standards

We are using standards developed by the Grid community and the National Science Foundation's Middleware Initiative (NMI). These standards create a software and protocol stack that is required to be on the Green Grid. Local to each node is a temporary scratch location which is published in a catalog. Nodes that have a shared file system (such as NFS or AFS) publish the location of a cluster-wide temporary file location. Some applications are installed into the Green Grid AFS Cell (grid.dartmouth.edu). This cell is currently available on all Green Grid systems. It is expected, however, that this will not be the case in the future. Users can submit jobs through the Globus GRAM interface using simple scripts or a DRM system such as PBS or Condor-G. Using wrapper scripts on top of these standards students are able to run many existing applications at a larger scale and with site independence.

4 Experiences

The concept of a campus-wide grid was conceived in May of 2004 and operational four months later (Fig 2). The Green Grid project started with the requirements from seven professors in four departments. With a demonstrated need for additional computational capacity we moved quickly to organize a proposal. We presented this proposal to the administration to seek support and the funds to purchase the hardware to bootstrap our Grid. We selected a platform which would provide participating groups compatibility with existing applications and high performance. The AMD Opteron architecture was selected for the 64-bit capabilities and price point. We solicited bids from four major system vendors and placed the order. Once the hardware began to arrive we started work on creating a single system image environment based on the Linux 2.4 kernel We constructed a website (http://grid.dartmouth.edu) for the distribution of Grid information, project descriptions, and procedures for using the Grid.

We have a small number of users from our academic departments utilizing the Green Grid for computational applications. These users, for the most part, have embarrassingly parallel or serial jobs to run. We have started computer science research into topics on Grid security, authorization, policy, and workflow. It is expected that the Green Grid

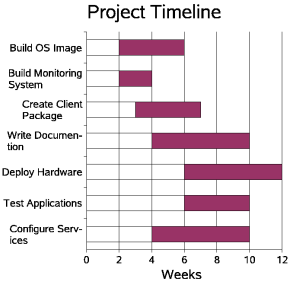


Fig. 2. Green Grid Implementation Schedule

Fig. 3. OCGE Grid Interface

could be used in undergraduate distributed systems classes and in laboratory science such as a fMRI data analysis offered by the Psychological and Brain Science Department. We have found some of the tools for establishing Grid credentials to not be either flexible enough or user friendly. Our current client software package includes a tool for certificate enrollment which has made this process easier. We have started to explore the use of a Grid Portal with the Open Grid Computing Environment (OGCE). We plan to have the portal available for users to manage their PKI credentials and Grid jobs (Fig 3).

5 The Intergrid

The Green Grid was designed to follow the model used on National Grids such as Grid3, the TeraGrid, and the EU-Data Grid. This design should provide for a trivial connection of Green Grid resources to larger Grid communities. A single department could, for example, join a science domain-specific Grid. The Green Grid can, as a whole, become a resource provider for a project such as the Open Science Grid[12].

The Green Grid's role in a federated Grid project such as the Open Science Grid is that of a resource provider. Dartmouth will have multiple virtual organizations (VO's) that will want to use resources available through OSG. We are participating in discussions on site and service agreements [13] that will provide policy statements on the integration of large campus grid with OSG. In addition to policy language and agreements there are technical issues around such topics as PKI cross-certification that will need to be worked out.

Recent work in the area of connecting Grids (e.g., [14, 15]) indicates that Bridge CAs can be used for Grid authentication across organizations. Jokl et al. found that two widely used Grid toolkits could be modified to support authentication with a Bridge

CA [15]. Their experiments used a testbed Bridge CA at the University of Virginia [14] with five large universities cross-certified to the Bridge CA.

6 Futures

The Green Grid currently has 120 processors on-line. It is shortly expected to grow to several hundred with the addition of Linux clusters in Computer Science, Research Computing, Dartmouth Medical School, and Psychological and Brain Sciences. The initial bootstrap infrastructure deployed in Phase I will be replaced as the systems are integrated in each department's local infrastructure. We expect several additional Grid services to appear on the Green Grid shortly including Replica Location Service (RLS) servers, high volume GridFTP servers, Virtual Data Catalogs (VDC), and the SHEMP proxy certificate management system. The OSG is due to come online in the spring of 2005 with Green Grid resources. Our work on this will provide an example for other institutions who wish to participate in this project.

Dartmouth is in the unique position of also operating the EDUCAUSE-chartered *Higher Education Bridge CA (HEBCA)*, which is intended to connect the PKIs of Higher Education institutions. Since the Grid community is about sharing resources, and HEBCA is positioned to enable PKI trust relationships between academic institutions, it seems like a natural evolution to use HEBCA to connect Grids in higher education.

Acknowledgments

We would like to thank the following people for their hard work, consultation, and patience in getting the Dartmouth Green Grid project off the ground:

Research Computing John Wallace, David Jewell, Gurcharan Khanna, and Richard Brittain
PKI Lab Kevin Mitcham and Robert J. Bentrup
Network Services Sean S. Dunten, Jason Jeffords, and Robert Johnson
Physics Bill Hamblen and Brian Chaboyer
Math Francois G. Dorais and Sarunas Burdulis
Computer Science Tim Tregubov and Wayne Cripps
Biology Mark McPeek
Tuck School of Business Geoff Bronner and Stan D. Pyc
Thayer School of Engineering Edmond Cooley
Dartmouth Medical School Jason Moore, Bill White, and Nate Barney
Dean of the Faculty of Arts & Sciences Michael S. Gazzaniga and Harini Mallikarach

We would like to also extend our thanks to Distributed Systems Lab at the University of Chicago and Argonne National Lab: Catalin L. Dumitrescu, Jens-S. Voeckler, Luiz Meyer, Yong Zhao, Mike Wilde, and Ian Foster.

James Dobson is supported in part by grants from the National Institutes of Health, NIH NS37470 and NS44393

References

[1] Ribbens, C.J., Kafura, D., Karnik, A., Lorch, M.: The Virginia Tech Computational Grid: A Research Agenda. Tr-02-31,, Virginia Tech (December 2002)
[2] The University of Michigan: MGRID (2004) http://www.mgrid.umich.edu.
[3] Global Grid Forum: The Global Grid Forum (2004) http://www.ggf.org.
[4] Pordes, R., Gardner, R.: The Grid2003 Production Grid: Principles and Practice. In: Thirteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC13). (2004)
[5] Johnston, W.E., Brooke, J.M., Butler, R., Foster, D.: Implementing Production Grids for Science and Engineering. In Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (2003)
[6] Novotny, J., Tuecke, S., Welch, V.: An online credential repository for the grid: Myproxy (2001)
[7] Marchesini, J.C., Smith, S.W.: SHEMP: Secure Hardware Enhanced MyProxy. Technical Report TR-2004-525, Computer Science Department, Dartmouth College (2004) http://www.cs.dartmouth.edu/ carlo/research/shemp/tr2004-525.pdf.
[8] OASIS: XACML 1.1 Specification Set. http://www.oasis-open.org (2003)
[9] Foster, I., Voeckler, J., Wilde, M., Zhao, Y.: The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration. In: First Biennial Conference on Innovative Data Systems Research. (2004)
[10] Avery, P., Foster, I.: The GriPhyN Project: Torwards Petascale Virtual Data Grids (2001) http://www.griphyn.org.
[11] Sehkri, V., Mandrichenko, I., Skow, D.: Site authorization service (SAZ). CoRR **cs.DC/0306100** (2003)
[12] The Open Science Grid Consortium: The Open Science Grid (2004) http://www.opensciencegrid.org.
[13] Open Science Grid Technical Policy Group: Open Science Grid Service Agreement Policy (2004)
[14] SURA: SURA NMI Testbed Grid PKI Bridge CA. https://www.pki.virginia.edu/nmi-bridge/ (2004)
[15] Jokl, J., Basney, J., Humphrey, M.: Experiences Using Bridge CAs for Grids. In: UK Workshop on Grid Security Experiences. (2004)

Resource-Aware Parallel Adaptive Computation for Clusters

James D. Teresco, Laura Effinger-Dean, and Arjun Sharma

Department of Computer Science, Williams College,
Williamstown, MA 01267 USA
terescoj@cs.williams.edu

Abstract. Smaller institutions can now maintain local cluster computing environments to support research and teaching in high-performance scientific computation. Researchers can develop, test, and run software on the local cluster and move later to larger clusters and supercomputers at an appropriate time. This presents challenges in the development of software that can be run efficiently on a range of computing environments from the (often heterogeneous) local clusters to the larger clusters and supercomputers. Meanwhile, the clusters are also valuable teaching resources. We describe the use of a heterogeneous cluster at Williams College and its role in the development of software to support scientific computation in such environments, including two summer research projects completed by Williams undergraduates.

Cluster computing environments at smaller institutions have provided a new platform for research and teaching in high-performance computing. Such local computing resources support development of software which can be executed on the local cluster or can be moved later to larger clusters or supercomputers for execution of larger problems. Meanwhile, clusters provide valuable local resources for teaching and the support of student projects. This paper describes a cluster at Williams College and provides an overview of a research effort that has been motivated and supported by this cluster, in particular two undergraduate projects which have contributed to this effort.

1 A Cluster Environment

Our cluster (known as "Bullpen"[1]) is located in the Department of Computer Science at Williams College. It consists of one Sun Enterprise 220R server with one 450MHz Sparc UltraII processor; two Enterprise 420R servers, each with four 450MHz Sparc UltraII processors; and six Enterprise 220R servers, each with two 450MHz Sparc UltraII processors; and four Sun Ultra 10 Workstations, each with one 300 or 333 MHz Sparc UltraII processor.

This cluster is intentionally heterogeneous, with its nodes having different processor speeds, numbers of processors and amount of memory per node. This

[1] http://bullpen.cs.williams.edu/

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 107–115, 2005.
© Springer-Verlag Berlin Heidelberg 2005

makes it an excellent platform for studies of scientific computation in heterogeneous and hierarchical environments. While most clusters are initially built using identical nodes, incremental growth is an attractive features of clusters. As new (likely faster) nodes are added, old nodes remain part of the cluster, leading to heterogeneity.

In addition to the support of the research described herein, this cluster has been used in Computer Science courses at Williams, most extensively in the Parallel Processing course. Students have been able to write multithreaded code using both POSIX threads [5] and OpenMP[2] to use the symmetric multiprocessing (SMP) nodes. They have used the Message Passing Interface (MPI)[3] to use multiple nodes to perform parallel computation with distributed memory and message passing. Student projects have included a parallel discrete event simulation, parallel particle simulations, a parallel photon mapper and a parallel ray tracer. Having the local cluster available meant that the students were not competing for processor cycles on lab workstations and did not have to develop software remotely at a supercomputing center.

2 Parallel Adaptive Computation on Clusters

Our goal is to develop tools and techniques to allow efficient parallel adaptive scientific computation on heterogeneous clusters such as Bullpen. We focus on solvers for systems of partial differential equations using finite element and related methods (e.g., [4, 6, 7]) that use meshes to discretize problem domains. The mesh is partitioned into subdomains consisting of disjoint subsets of mesh entities (e.g., elements, surfaces, nodes) and these subdomains are assigned to the cooperating processes of a parallel computation. Adjacent mesh entities will exchange information during the solution process. So in addition to its attempts to divide the work evenly among the processes (to achieve load balance), a mesh partitioner attempts to minimize the number of pairs of adjacent entities which are assigned to different processes (to minimize interprocess communication).

The methods are adaptive, where time and space efficiency is improved by concentrating computational effort in parts of the domain where it is needed to achieve a solution to a prescribed accuracy [1]. However, adaptivity will disturb a balanced partitioning, necessitating a dynamic load balancing step. Dynamic load balancing procedures have similar goals to mesh partitioners, but must operate on already-distributed data and should minimize the change between the existing decomposition and the new decomposition (to limit mesh migration).

A number of approaches to dynamic load balancing have been proposed ([8] includes a survey). The Zoltan Parallel Data Services Toolkit [2] provides a common interface to high-quality implementations of several such procedures. With Zoltan, applications can quickly make use of and can easily switch among available load balancing methods. Fig. 1 shows the interaction between parallel

[2] `http://www.openmp.org`
[3] `http://www-unix.mcs.anl.gov/mpi/`

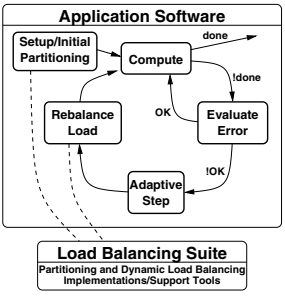


Fig. 1. Program flow of a typical parallel adaptive computation using a load balancing suite such as Zoltan

adaptive application software and a dynamic load balancing suite such as that in Zoltan. After an initial partitioning, the application performs computation steps, periodically evaluating error estimates and checking against specified error tolerances. If the error is within tolerance, the computation continues. Otherwise, an adaptive refinement takes place, followed by dynamic load balancing before the computation resumes.

Our goal is to run parallel adaptive computations efficiently on heterogeneous clusters, while making minimal changes to the application software. We have been working with three software packages in cluster environments. *LOCO* [4] and *DG* [7] implement parallel adaptive discontinuous Galerkin procedures. The Parallel Hierarchical Adaptive MultiLevel software (PHAML) [6] implements a variety of parallel adaptive solution procedures. Each of these uses Zoltan's dynamic load balancing procedures.

3 Resource-Aware Load Balancing

In cluster environments, load imbalance may be introduced because of heterogeneous or non-dedicated processors. The relative costs of computation and communication may change from one environment to the next, suggesting a different partitioning strategy. On Bullpen, we are faced with nonuniform processor speeds, the mixture of 1-, 2-, and 4-processor nodes, and a slower network relative to processing speed than previous target platforms. A *resource-aware* computation, which requires knowledge of the computing environment and tools to make use of this knowledge, is needed to take full advantage of the computing environment. Resource-aware adjustments can be made anywhere from low-level tools to application programs (see [11] for examples).

Our focus is on resource-aware dynamic load balancing, guided by the Dynamic Resource Utilization Model (DRUM) [3, 10][4]. Processor "speed" (mega-

[4] DRUM web page: `http://www.cs.williams.edu/drum/`

hertz or gigahertz) ratings must be combined with other factors such as cache, memory and input/output subsystem performance, and current usage to determine how quickly a processor can perform computation. DRUM evaluates the computing environment's performance using data from both benchmarks which are run *a priori* either manually or from within DRUM's graphical configuration tool (Section 4) and a dynamic performance monitors. DRUM distills this information into a single "power" value, readily used by load balancing procedures (including all Zoltan procedures) to produce appropriately-sized partitions.

Benchmark results are stored in a model of the computing environment that encapsulates information about hardware resources, their capabilities and their interconnection topology in a tree structure. The root of the tree represents the total execution environment. The children of the root node are high level divisions of different networks connected to form the total execution environment. Sub-environments are recursively divided, according to the network hierarchy, with the tree leaves being individual single-processor (SP) nodes or symmetric multiprocessing (SMP) nodes. *Computation nodes* at the leaves of the tree have data representing their relative computing and communication power. *Network nodes*, representing routers or switches, have an aggregate power calculated as a function of the powers of their children and the network characteristics.

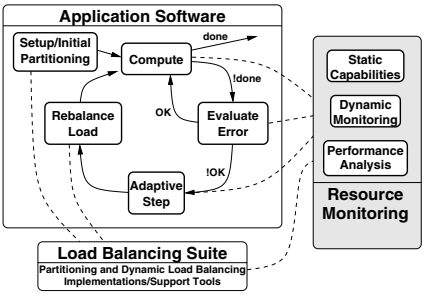


Fig. 2. A typical interaction between an adaptive application code and a dynamic load balancing suite, when using a resource monitoring system (e.g., DRUM)

DRUM also provides a mechanism for dynamic monitoring and performance analysis. Monitoring agents in DRUM are threads that run concurrently with the user application to collect memory, network, and CPU utilization and availability statistics. Fig. 2 shows the interaction among an application code, a load balancing suite such as Zoltan, and a resource monitoring system such as DRUM for a typical adaptive computation. When load balancing is requested, the load balancer queries the monitoring system's performance analysis component to determine appropriate parameters and partition sizes for the rebalancing step.

DRUM can also adjust for heterogenous, hierarchical, and non-dedicated network resources by estimating a node's communication power based on the communication traffic at the node. Information about network interfaces may

be gathered using kernel statistics, a more portable but still limited library called `net-snmp`[5], which implements the Simple Network Management Protocol (SNMP), or the Network Weather Service (NWS) [13] (Section 5). Giving more work to a node with a larger communication power can take advantage of the fact that it is less busy with communication, so should be able to perform some extra computation while other nodes are in their communication phase. The communication power is combined with processing power as a weighted sum to obtain the single value that can be used to request appropriately-sized partitions from the load balancer.

We have used DRUM to guide resource-aware load balancing for both the PHAML and DG application software. DRUM-guided partitioning shows significant benefits over uniformly sized partitions, approaching, in many instances, the optimal relative change in execution times. We have also seen that DRUM can effectively adjust to dynamic changes, such as shared use of some nodes. This cannot be done with a static model that takes into account only node capabilities. Our focus in this paper is on the two DRUM enhancements described in the following sections; see [3] and [10] for performance studies using DRUM.

4 A Graphical Configuration Tool for DRUM

DRUM constructs its run-time model of the computing environment using information stored in an XML-format configuration file that describes properties of the system (e.g., benchmark results, network topology). We have developed a graphical configuration program in Java called *DrumHead* that aids in the construction of these configuration files[6]. DrumHead can be used to draw a description of a cluster, automatically run the benchmarks on the cluster nodes, and then create the configuration file for DRUM to read in when constructing its model. Fig. 3 shows an excerpt from an XML configuration file generated by DrumHead for the Bullpen Cluster configuration.

The layout of the main window (Fig. 4) is simple: a panel of tools and buttons on the left and a workspace (starting out empty) on the right. The tool pane shows the current properties of the entire cluster, all the changeable features of the selected node and buttons to save changes to the selected node's parameters. In the middle pane, the user can draw nodes, represented by rectangles (the computing nodes) and ovals (networking nodes), connected by lines. These nodes can be dragged, so the user can place them in a meaningful arrangement.

DrumHead allows specification of dynamic load balancing methods and parameters for network and SMP computing nodes. These parameters can be used by DRUM to guide a hierarchical load balancing, where different load balancing procedures are used in different parts of the computing environment. The available procedures present tradeoffs in execution time and partition quality (*e.g.*,

[5] `http://www.net-snmp.org`

[6] The design and implementation of DrumHead was part of the research project of Williams undergraduate Arjun Sharma during Summer 2004.

```
<machinemodel>
<node type="NETWORK" nodenum="0" name="" IP="" isMonitorable="false"
parent="-1" imgx="361.0" imgy="52.0">
<lbmethod lbm="HSFC" KEEP_CUTS="1"></lbmethod></node>
<node type="SINGLE_COMPUTING" nodenum="2" name="mendoza.cs.williams.edu"
IP="137.165.8.140"  isMonitorable="true" parent="0"
benchmark="52.43" imgx="50.0" imgy="138.0"></node>
<node type="MULTIPLE_COMPUTING" nodenum="3" name="rivera.cs.williams.edu"
IP="137.165.8.130" isMonitorable="false" parent="0"
imgx="74.64 "imgy="263.0" benchmark="82.55" numprocs="4">
<lbmethod lbm="HSFC" KEEP_CUTS="1">
</lbmethod></node>
...
</machinemodel>
```

Fig. 3. An excerpt from a configuration file generated by DrumHead for Bullpen

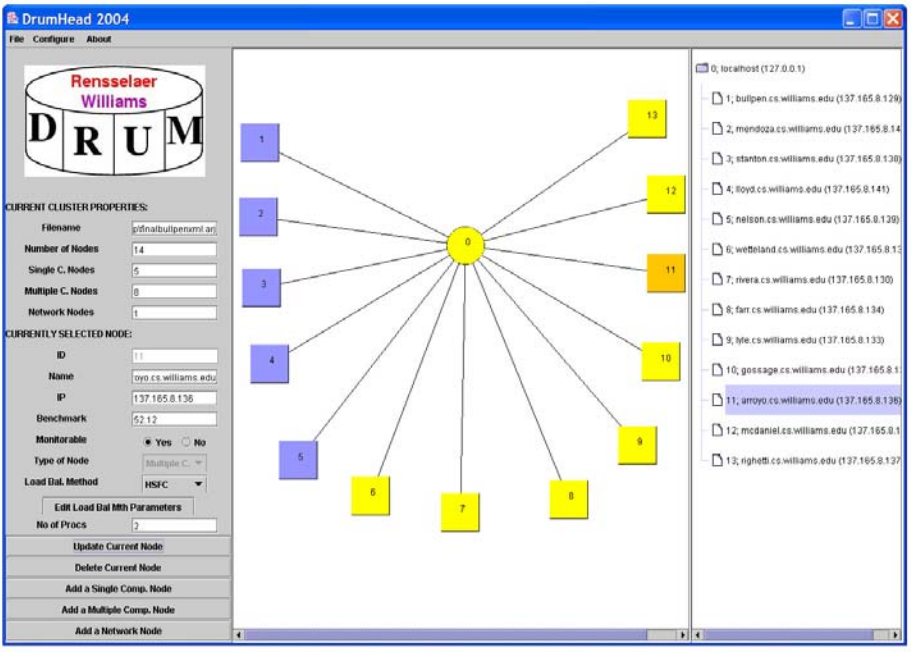


Fig. 4. DrumHead editing a description of the Bullpen Cluster

surface indices, interprocess connectivity, strictness of load balance) [12] and some may be more important than others in some circumstances. For example, consider a run using two or more of Bullpen's SMP nodes. A more costly graph partitioning can be done to partition among the SMPs, to minimize communication across the slow network interface, possibly at the expense of some com-

putational imbalance. Then, a fast geometric algorithm can be used to partition independently within each node. Hierarchical balancing, which is implemented in Zoltan, is described in detail in [9].

5 Interface to the Network Weather Service

DRUM is intended to work on a variety of architectures and operating systems. We do not want to require that DRUM users install additional software packages, but we do want DRUM to take advantage of such packages when available. We have developed an interface that allows DRUM to access information from NWS[7], which provides information about network and CPU usage for Unix-based systems. NWS is more intrusive than DRUM's other network monitoring capabilties, as it will send its own messages to measure network status.

NWS uses a set of "sensor" servers which run separately on each node of the parallel system, interacting with a "nameserver" and one or more "memory" servers. The nameserver allows easy searching for servers ("hosts"), sensor resources ("activities" or "skills"), and previously-collected data ("series"). For instance, to search for statistics about bandwidth between machine **A** and machine **B**, you would query the nameserver for an object with properties `objectClass` "nwsSeries," `resource` "bandwidthTcp," `host` "A:8060," and `target` "B:8060," where 8060 is the port used by the sensors on **A** and **B**. Network data is gathered within "cliques" of nodes: sets of machines which trade packets to measure bandwidth, connect time, and latency. The concept of cliques fits well with DRUM's tree model, as a clique may be defined as all leaves of a network node.

DRUM relies on the user or system administrator to configure and launch the appropriate NWS servers on each node within the parallel system. NWS activities could be started from within DRUM, but this would be ineffective early in a computation as NWS needs at least a few minutes to collect enough data to provide useful information. When it needs to gather network statistics from NWS, DRUM searches the nameserver for available "bandwidthTcp" series and randomly selects three. These series are limited to those whose host is the current machine and whose target shares a parent node with the host. From these three series, DRUM calculates the communication power of the node based one of three methods: an average of 20 measurements, the most recent single measurement, or an NWS "forecast," which essentially provides a normalized estimate of bandwidth, undisturbed by small variations. This bandwidth calculation substitutes for the "communication activity factor" used by the kstat- and SNMP-based implementations for the determination of communication powers and weights in DRUM's overall power formulas [3].

[7] The implementation of the DRUM interface to NWS was part of the research project of Williams undergraduate Laura Effinger-Dean during Summer 2004.

Acknowledgments

Teresco was supported in part by Sandia contract PO15162 and the Computer Science Research Institute at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000. Effinger-Dean and Sharma were supported by the Williams College Summer Science Research program. DRUM was developed with Jamal Faik (Rensselaer). Erik Boman, Karen Devine, and Bruce Hendrickson (Sandia) and Luis Gervasio (Rensselaer) also contributed to the design of DRUM.

References

1. K. Clark, J. E. Flaherty, and M. S. Shephard. *Appl. Numer. Math., special ed. on Adaptive Methods for Partial Differential Equations*, 14, 1994.
2. K. Devine, E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.
3. J. Faik, J. D. Teresco, K. D. Devine, J. E. Flaherty, and L. G. Gervasio. A model for resource-aware load balancing on heterogeneous clusters. Technical Report CS-05-01, Williams College Department of Computer Science, 2005. Submitted to Transactions on Parallel and Distributed Systems.
4. J. E. Flaherty, R. M. Loy, M. S. Shephard, and J. D. Teresco. Software for the parallel adaptive solution of conservation laws by discontinuous Galerkin methods. In B. Cockburn, G. Karniadakis, and S.-W. Shu, editors, *Discontinous Galerkin Methods Theory, Computation and Applications*, volume 11 of *Lecture Notes in Compuational Science and Engineering*, pages 113–124, Berlin, 2000. Springer.
5. B. Lewis and D. J. Berg. *Multithreaded Programming with pthreads*. Sun Microsystems Press, 1997.
6. W. F. Mitchell. The design of a parallel adaptive multi-level code in Fortran 90. In *International Conference on Computational Science (3)*, volume 2331 of *Lecture Notes in Computer Science*, pages 672–680. Springer, 2002.
7. J.-F. Remacle, J. Flaherty, and M. Shephard. An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems. *SIAM Review*, 45(1):53–72, 2003.
8. J. D. Teresco, K. D. Devine, and J. E. Flaherty. *Numerical Solution of Partial Differential Equations on Parallel Computers*, chapter Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations. Springer-Verlag, 2005.
9. J. D. Teresco, J. Faik, and J. E. Flaherty. Hierarchical partitioning and dynamic load balancing for scientific computation. Technical Report CS-04-04, Williams College Department of Computer Science, 2004. Submitted to Proc. PARA '04.
10. J. D. Teresco, J. Faik, and J. E. Flaherty. Resource-aware scientific computation on a heterogeneous cluster. Technical Report CS-04-10, Williams College Department of Computer Science, 2005. To appear, *Computing in Science & Engineering*.
11. J. D. Teresco, J. E. Flaherty, S. B. Baden, J. Faik, S. Lacour, M. Parashar, V. E. Taylor, and C. A. Varela. Approaches to architecture-aware parallel scientific computation. Technical Report CS-04-09, Williams College Department of Computer Science, 2005. Submitted to *Proc. PP'04: Frontiers of Scientific Computing*.

12. J. D. Teresco and L. P. Ungar. A comparison of Zoltan dynamic load balancers for adaptive computation. Technical Report CS-03-02, Williams College Department of Computer Science, 2003. Presented at COMPLAS '03.
13. R. Wolski, N. T. Spring, and J. Hayes. The Network Weather Service: A distributed resource performance forecasting service for metacomputing. *Future Generation Comput. Syst.*, 15(5-6):757–768, October 1999.

New Algorithms for Performance Trace Analysis Based on Compressed Complete Call Graphs

Andreas Knüpfer and Wolfgang E. Nagel

Center for High Performance Computing,
Dresden University of Technology, Germany
{knuepfer, nagel}@zhr.tu-dresden.de

Abstract This paper addresses performance and scalability issues of state-of-the-art trace analysis. The Complete Call Graph (CCG) data structure is proposed as an alternative to the common linear storage schemes. By transparent in-memory compression CCGs are capable of exploiting redundancy as frequently found in traces and thus reduce the memory requirements notably. Evaluation algorithms can be designed to take advantage of CCGs, too, such that the computational effort is reduced in the same order of magnitude as the memory requirements.

1 Introduction

Todays High Performance Computing (HPC) is widely dominated by massive parallel computation, using very fast processors [1]. HPC performance analysis and particularly tracing approaches are affected by that trend. The evolution of computing performance combined with more advanced monitoring and tracing techniques lead to very huge amounts of trace data. This is becoming a major challenge for trace analysis - for interactive investigation as well as for automatic analysis. With interactive work flows the requirement for fast response times is most important for analysis tools. For automatic or semi-automatic tools that use more or less computational expensive algorithms and heuristics this is a severe problem, too. Both have in common that the effort depends on the amount of trace data at least in a linear way.

The state of the art in-memory data structures for trace data suggest linear storage only, i.e. arrays or linear linked lists [2, 11, 4, 14]. Even though they are fast and effective for raw access, they lack opportunities for improvement.

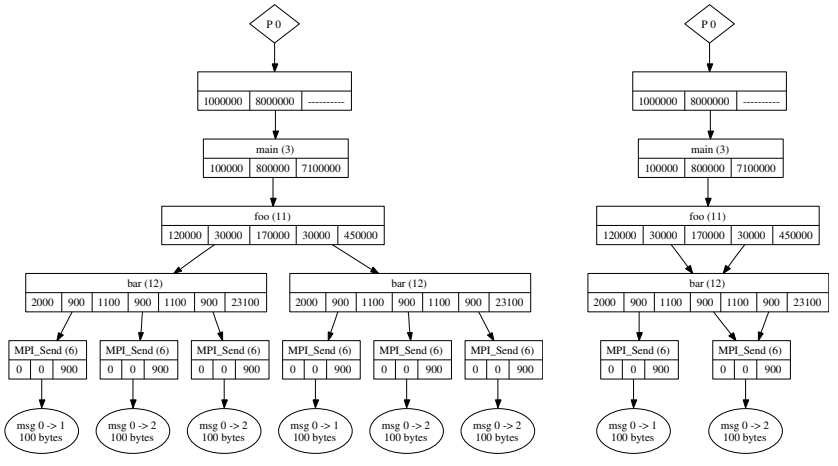
The Compressed Complete Call Graph (cCCG) is a promising alternative approach for storing trace data. Contrary to linear data structures it offers fast hierarchical access. Furthermore, it supports transparent compression of trace data which saves memory consumption as well as computational effort for some kinds of queries.

The following Section 2 gives a concise overview and definition of cCCGs as well as its properties. Based on that, the concept of Cached Recursive Queries onto cCCGs is developed in Section 3. Also, it provides some typical examples of use for this algorithm with performance results. Finally, Section 4 concludes the paper and gives an outlook on future work.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 116–123, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Compressed Complete Call Graphs (cCCGs)

An alternative way to the traditional linear scheme is to re-create the complete function call tree. This preserves the temporal order as well as the call hierarchy. A Complete Call Tree (CCT) contains the comprehensive function call history of a process. This makes it different from ordinary Call Trees which store a summarized caller to callee relation only [6]. Furthermore, not only function call events but also all other kinds of events can be contained such as message send/receive events, I/O events or hardware performance counter samples. However, the function call hierarchy determines the structure of the CCT. Figure 1(a) shows an example of a Complete Call Tree including some `MPI_Send()` calls.



(a) uncompressed (b) compressed

Fig. 1. An example Complete Call Graph (a) and its compressed counterpart (b)

This figure shows also another difference to the traditional way of storing trace data. Instead of time stamps the CCT holds time durations which is most important for the compression part. Of course, the back and forth transformation between time stamps and time durations is simple. In terms of expressiveness a CCT is equivalent to a traditional linear data structure with time stamps.

2.1 Compression

The more structured CCT representation makes it easy to detect identical nodes and sub-trees of pairwise identical nodes. All groups identical sub-trees are then replaced by references to a single representative item. All other instances can be eliminated completely. This removal of redundancy is a typical strategy for data compression and transforms CCTs to Compressed Complete Call Graphs (cCCGs). Figure 1(b) shows the compressed counterpart of the previously mentioned CCT in Figure 1(a). Of course, this transformation destroys an essential

property of tree graphs, namely the single parent node property. However, all construction and query algorithms can be implemented in a way not to rely on that. Thus, this kind of compression can be said to be completely transparent with respect to read access.

So far, cCCGs offer a completely lossless compression scheme for trace data. HPC programs show a high degree of repetition and regularity. This is reflected in traces as well. Therefore, this compression strategy works reasonably well.

At this point, it is possible to allow not only equal but also *similar* subtrees to be mapped onto one another. This takes this approach another step further introducing lossy compression. However, this is applicable for selected members of the node data structure only. For example, identifiers for functions or processes must not be subject to lossy compression because this would render the information invalid. Other data members as time durations, operation count, message volumes etc. are robust against small deviations in the data.

So, all those thoughts need to be considered when defining what *similar* is supposed to mean for sub-graphs. Furthermore, all deviations introduced must be limited by selectable bounds. This will induce error bounds for all results computed from data carrying deviations.

Following this construction scheme plainly there arises one major disadvantage in terms of general graph data structures. As the structure of a CCG is determined by the call hierarchy alone, the tree's branching factor is unbounded and probably very large. This causes two negative effects. First, large branching factors are most undesirable for tree traversal algorithms. Second, the compression ability is enhanced by rather small branching factors By introducing special nodes the branching factor can be bounded to an arbitrary constant ≥ 2 [7].

2.2 Compression Metrics

In order to make the compression comparable a measure for compression is needed. For cCCGs there are two metrics suitable for different purposes:

$$R_m := \frac{\text{Memory}_0}{\text{Memory}_{compressed}} \ , \ R_n := \frac{\text{Nodes}_0}{\text{Nodes}_{compressed}} = \frac{N}{n} \tag{1}$$

First, the ratio R_m of the raw memory consumption of graph nodes including references (pointers) to child nodes is suitable for estimating memory consumption. This is a the key issue as soon as data compression is concerned. Second, the node count ratio R_n is very useful when estimating the computational effort for tasks that perform constant amount of work per graph node. Since single nodes have variable memory footprints R_n is not proportional to R_m.

Practical experiments with real world traces from 20 MB up to 2 GB have shown very promising results [7]. For zero time deviation bounds R_m ranges from 2 to 8 and R_n lies in between 5 and 14. For large traces with midrange deviation bounds for time information of 1000 ticks (timer units) or 50 % the memory compression ratio R_m rises up to 44 while the node compression ratio R_n climbs up to 93. With more relaxed bounds R_m and R_n rise over 1000!

Compression ratios of $R_X < 1$ are impossible, and the memory requirements for uncompressed CCGs and traditional linear data structures are about the

same. In general, it can be stated that more relaxed error bounds lead to better compression. Furthermore, larger traces usually yield better compression than shorter ones. Moreover, the higher the final compression ratio will grow, the faster the compression itself will be. Within the CCG creation algorithm the construction and compression steps are closely integrated such that at no point the whole uncompressed graph needs to be stored. The overall complexity for cCCG construction is $O(N \cdot m)$ with the node count in the uncompressed CCG N and a rather small factor m. For construction, split and compression algorithms, complexity analysis and experimental performance results see [10].

3 Cached Recursive Queries

After creation from trace or re-creation from a previously saved version the Compressed Complete Call Graph is available in main memory for querying. This might involve automatic evaluation procedures or interactive user queries. This article focuses on interactive queries particularly with regard to visualization tasks while, of course, the cCCG data structure is suitable for performing automatic analysis, too.

One of the two most important kinds of queries is the so called *Summary Query*. It computes a certain property for a given time interval and a given set of processes. Typical examples for summary queries are exclusive or inclusive run time per function, message volumes per pairs of processes, average hardware performance counter values and many more.

The traditional algorithm to derive summary statistics performs a single linear read-through of all process traces and uses some additional temporary memory to re-play the call stack. While this read-through in temporal order can be emulated by cCCGs another algorithm is proposed here. Following the tree-like graph structure a recursive traversal of the graph seems most obvious. This is well suited to calculate the query's result in a divide and conquer style. Considering an uncompressed CCG the graph itself is its own evaluation graph as shown in Figure 2 for a single process. From the computational complexity point of view this algorithm is equal to the traditional way with $O(N)$ effort.

For successive queries with overlapping time intervals this evaluation scheme can be improved by caching of intermediate results at graph nodes. Especially for interactive navigation within a trace such sequences of successive queries with non-disjoint time intervals are very common. Most of the time, it involves an initial global query followed by multistage zooming into some interesting regions for closer examination.

Caching avoids re-computation of intermediate results that appear multiple times. That means, whenever the evaluation encounters existing cached results the computation graph is pruned, i.e. the underlying sub-tree must not be traversed. See Figure 2 for an illustrated example.

Typical cache strategies like Most Frequently Used (MFU) or Most Recently Used (MRU) are not feasible here, assuming that the cache is small in comparison to node count. When inserting newly computed results this would lead to

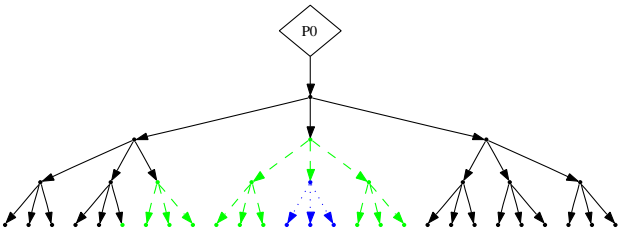


Fig. 2. Evaluation graph for querying a uncompressed CCG. This is identical to the CCGs own structure. Marked green (dashed) and blue (dotted) there are two successive nested queries to the initial global query. For both the recursion can be pruned as soon as a cached intermediate result is found for one of the graph nodes

continuous cache thrashing. Instead, heuristics are utilized to limit the number of entries. For example, one could insert every n'th result into the cache. Another convenient strategy is to insert only nodes of certain depth levels d_{node} with

$$d_{node} \text{ modulo } x = y, \;\; y < x. \tag{2}$$

The latter strategy allows to estimate the computational effort, because there are at maximum x depth levels to traverse before there are cached results for all nodes. With maximum branching factor b this yields worst case effort of $O(b^x)$.

In addition to this effort, there is a preceding effort of finding the smallest graph node containing the given time interval. This involves following a single path from the root node downwards if maximum length d which is the maximum tree depth. Furthermore, any result for nodes intersecting the bounds of the current time interval cannot be cached but must be computed in a special way which is aware of the interval bound. Therefore, all nodes intersecting the time interval bounds must be handled separately. For both interval bounds this involves at maximum $2 \cdot d$ nodes. Thus, the overall complexity is $O(d + b^x) \leq O(N)$.

3.1 Application to Compressed CCGs

This algorithm can be applied to compressed CCGs as well. Actually, compression contributes to a tremendous improvement of this evaluation scheme. Since graph nodes are referenced repeatedly, caching pays off even for initial queries. In addition, the compressed node count $n \ll N$ causes a reduced cache size. This saving is proportional to the node compression ratio $R_n = \frac{N}{n}$. At the same time, the cache re-use frequency is increased by that factor. Alternatively, the parameters for the caching heuristics can be relaxed - compare Equation (2). Figure 3 gives an impression how the compressed counterpart to the example in Figure 2 might look like and how initial and successice queries are performed.

The second of the two most important kinds of interactive queries is the Timeline Visualization Query. It shows the behavior of some properties over the course of time representing values with a color scale. Subject of timeline displays might again be statistical values like run time share, communication volumes or

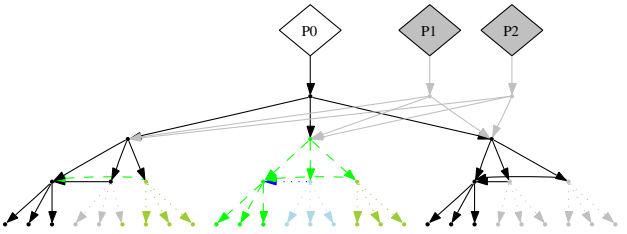


Fig. 3. Evaluation graph of successive queries on a compressed CCG. Just like within the CCG itself some sub-trees are replaced by references to other instances. Thus, some sub-trees are referenced more than once. Intermediate results for such sub-trees can be re-used within the initial query and in all successsice queries. Sub-trees might even be shared by multiple processes, e.g. processes P1 and P2 might references sub-trees originally from P0

hardware performance counter values. It might also be the characterization of the function call behavior, maybe even including the call hierarchy. This is in fact the most commonly used variety of timeline displays [3, 2].

Always, timeline visualizations are rendered for a given horizontal display resolution of $w \ll N$ pixels. With traditional data structures this requires linear effort $O(N)$ for a single read-through at least. Based on cCCGs, this quite different task can be converted to a series of Cached Recursive Queries, each one restricted to the time interval associated with a single pixel. This allows to transfer the reduced effort algorithm (with $O(d + b^x)$) to this problem, too.

3.2 Performance Results

After the theoretical considerations some performance results shall emphasize the practical advantages of the cCCG based Cached Recursive Queries. As test candidate a trace file from the ASCI benchmark application IRS [13] was used which is 4 GB in size (in VTF3 format [12]) and contains 177 million events. This measurements were performed on an AMD Athlon 64 workstation with 2200 MHz speed and 2 GB of main memory.

Figure 4 shows the evaluation time for a Cached Recursive Query computing exclusive and inclusive time as well as occurrences count per function all at once. It is presented depending on the node compression ratio R_n of the cCCG and the cache strategy parameter x as in Equation (2). The left hand side shows initial queries which take $1s$ to $23s$ depending on compression rate. There is only a minor influence of the caching parameter. On the right hand side, run times for successive queries are shown, again with global scope. Here, the run time ranges from $50ms$ to $400ms$ which is without doubt suitable for truly interactive responses. For non-global queries restricted to a smaller time interval both, initial and successive queries will be even faster.

In comparison to traditional evaluation on linear data structures this is an important improvement. For example, the classical and well known trace analysis tool Vampir [2] is incapable of providing any information about the example

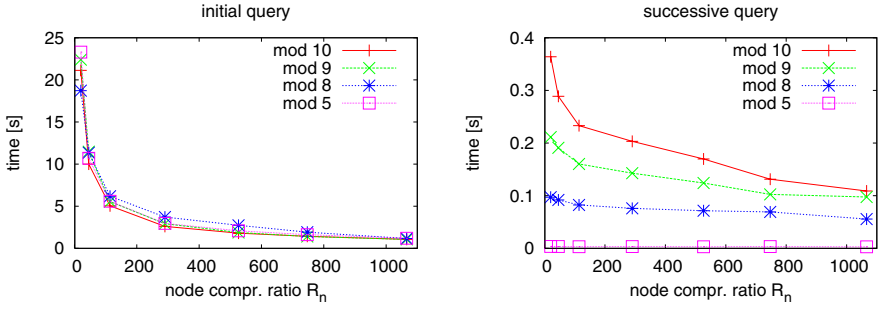


Fig. 4. Run time for initial (left) and successive (right) global Cached Recursive Queries on cCCGs with different compression levels and caching parameters

trace just because of its size on the given workstation as 2 GB main memory are insufficient to contain the trace. Furthermore, on any 32 bit platform the address range is not large enough. Thus, the new approach is not only a an improvement in speed but also in receiving any valid information or not.

4 Conclusion and Outlook

The paper presented a novel evaluation algorithm for Compressed Complete Call Graphs. This Cached Recursive Query is capable of delivering results in a truly interactive fashion even for larger traces. This is especially necessary for convenient manual analysis and navigation in traces. For large traces, this is superior to the traditional scheme. Furthermore, this algorithm unites the tasks of computing statistical summary queries and of generating timeline diagrams.

Parallelizing and distributing the Compressed Complete Call Graph approach and the Cached Recursive Query algorithm is another option to extend the range of manageable trace file sizes. This has already been implemented in a successful experiment [8] introducing the cCCG data structure into Vampir NG [5].

Future research will focus on automatic and semi-automatic performance analysis techniques based on cCCG data structures. First, this aims at applying known procedures to cCCGs taking advantage of compression and reduced memory footprint. Second, this extends to developing new methods. Some result have already been published in [9].

References

1. George Almasi, Charles Archer, John Gunnels, Phillip Heidelberger, Xavier Martorell, and Jose E. Moreira. Architecture and Performance of the BlueGene/L Message Layer. In *Recent Advances in PVM and MPI. Proceedings of 11th European PVM/MPI Users Group Meeting*, volume 3241 of *Springer LNCS*, pages 259–267, Budapest, Hungary, September 2004.

2. H. Brunst, H.-Ch. Hoppe, W.E. Nagel, and M. Winkler. Performance Otimization for Large Scale Computing: The Scalable VAMPIR Approach. In *Proceedings of ICCS2001, San Francisco, USA*, volume 2074 of *Springer LNCS*, page 751. Springer-Verlag Berlin Heidelberg New York, May 2001.
3. H. Brunst, W. E. Nagel, and S. Seidl. Performance Tuning on Parallel Systems: All Problems Solved? In *Proceedings of PARA2000 - Workshop on Applied Parallel Computing*, volume 1947 of *LNCS*, pages 279–287. Springer-Verlag Berlin Heidelberg New York, June 2000.
4. Holger Brunst, Allen D. Malony, Sameer S. Shende, and Robert Bell. Online Remote Trace Analysis of Parallel Applications on High-Performance Clusters. In *Proceedings of ISHPC'03 Conference*, volume 2858 of *Springer LNCS*, pages 440–449, 2003.
5. Holger Brunst, Wolfgang E. Nagel, and Allen D. Malony. A Distributed Performance Analysis Architecture for Clusters. In *IEEE International Conference on Cluster Computing, Cluster 2003*, pages 73–81, Hong Kong, China, December 2003. IEEE Computer Society.
6. David Grove and Craig Chambers. An assessment of call graph construction algorithms. `http://citeseer.nj.nec.com/grove00assessment.html`, 2000.
7. Andreas Knüpfer. A New Data Compression Technique for Event Based Program Traces. In *Proccedings of ICCS 2003 in Melbourne/Australia, Springer LNCS 2659*, pages 956 – 965. Springer, Heidelberg, June 2003.
8. Andreas Knüpfer, Holger Brunst, and Wolfgang E. Nagel. High Performance Event Trace Visualization. In *13th Euromicro Conference on Parallel, Distributed and Network-based Processing*, Lugano, Switzerland, Feb 2005.
9. Andreas Knüpfer, Dieter Kranzlmüller, and Wolfgang E. Nagel. Detection of Collective MPI Operation Patterns. In *Recent Advances in PVM and MPI. Proceedings of 11th European PVM/MPI Users Group Meeting*, volume LNCS 3241, pages 259–267, Budapest, Hungary, September 2004. Springer.
10. Andreas Knüpfer and Wolfgang E. Nagel. Compressible Memory Data Structures for Event Based Trace Analysis. *Future Generation Computer Systems by Elsevier*, 2004. [accepted for publication].
11. Dieter Kranzlmüller, Michael Scarpa, and Jens Volkert. DeWiz - A Modular Tool Architecture for Parallel Program Analysis. In *Euro-Par 2003 Parallel Processing*, volume 2790 of *Springer LNCS*, pages 74–80, Klagenfurt, Austria, August 2003.
12. S. Seidl. VTF3 - A Fast Vampir Trace File Low-Level Library. personal communications, May 2002.
13. The ASCI Project. The IRS Benchmark Code: Implicit Radiation Solver. `http://www.llnl.gov/asci/purple/benchmarks/limited/irs/`, 2003.
14. F. Wolf and B. Mohr. EARL - A Programmable and Extensible Toolkit for Analyzing Event Traces of Message Passing Programs. Technical report, Research Center Jülich, April 1998. FZJ-ZAM-IB-9803.

PARADIS: Analysis of Transaction-Based Applications in Distributed Environments

Christian Glasner[1], Edith Spiegl[1], and Jens Volkert[1,2]

[1] Research Studios Austria, Studio AdVISION,
Leopoldskronstr. 30, 5020 Salzburg, Austria
{christian.glasner, edith.spiegl}@researchstudio.at
http://www.researchstudio.at/advision.php
[2] GUP - Institute of Graphics and Parallel Processing,
Joh. Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria
volkert@gup.uni-linz.ac.at
http://www.gup.uni-linz.ac.at

Abstract. The analysis of long running and distributed applications poses a great challenge to software developers. PARADIS is a novel tool that helps the programmer with accomplishing this task. It reconstructs the corresponding event graph from events collected during a program run and provides techniques to address the problems arising from large traces. It offers several modules for specific examinations like the analysis of applications which process transactions and due to its modular architecture it allows an easy extension of the functionality. We show the usefulness on the basis of a real-life application and discuss future enhancements.

1 Introduction

Program analysis and debugging are complex tasks in the area of software engineering. Using profilers is a common way to understand the dynamic behavior of an application, since they allow measuring the time being spent in particular functions and consequently help to identify possible performance bottlenecks. Unfortunately there are lots of reasons for bad runtime behavior that cannot be tracked by this technique because simple time measurement only shows which functions were responsible for the execution time but not the underlying causes.

This reason led to event-based debugging. The programmer or the tool automatically instruments an application at arbitrary points and at these points information about state changes happening during a program run are recorded. Each state record is associated with an event, that triggered the logging activity and a series of recorded events is called a trace. These traces can be analyzed either after termination of the inspected application (post-mortem) or simultaneously (on-line). In practice this event-based approach is limited by the number of events which have to be gathered. If a high number of events are recorded it not

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 124–131, 2005.
© Springer-Verlag Berlin Heidelberg 2005

only slows down the execution of the program because of the logging activities (if not avoided by additional hardware), but it also complicates the later analysis. Possible problems arising are the time spent for trace processing and analysis, the consumption of disk storage and working memory and the complexity of the graphical representation.

To keep the number of events manageable one can instrument the program very economically. However, trends show that for utilizing the capacity of all available resources more and more applications get distributed across multiple threads, processes or even nodes. This leads to higher complexity of the applications and to new sources of error. Taking into consideration multiprocessor machines, clusters or grids combining hundreds or thousands of processors, the need for detailed information about the program execution to find out the reasons for an application's unintentional behavior seems obvious. Even if enough program information is gathered and sufficient computing power is provided one still has to face the task of filtering the data to get valuable results during the analysis and visualization.

In this paper we present PARADIS, a tool for the event-based analysis of distributed programs. In Section 2 we discuss related work, while Section 3 focuses on the modular architecture of the tool. Section 4 describes a real world example and finally an outlook on future work concludes the paper.

2 Related Work

There are several tools that address performance analysis of parallel programs. According to the programming paradigm of the underlying program, they log communication events like Send and Receive or resource assignment in shared memory systems. They deal with large trace files but do not offer support for the analysis of transaction-based applications, where events happening in the context of a single transaction belong semantically together.

Paradyn [1] for instance, which was developed at the University of Wisconsin, uses dynamic instrumentation to gather performance data such as CPU times, wallclock times, and relevant quantities for I/O, communication, and synchronization operations. As it allows dynamically setting and removing predefined probes during a program's execution the amount of generated trace data can be kept relatively small, even when monitoring over a long period.

Vampir [2] and Vampir NG [3] analyze trace files produced by the Vampirtrace library which has an API for defining events. Instrumentation is realized by linking the application with the library, after adding the calls to Vampirtrace to the source code. Similar to PARADIS they offer a hierarchical visualization, but as the application is targeted at clustered SMP nodes, the display provides three dedicated layers that represent cluster, nodes and processes [4].

Alike PARADIS DeWiz [5] utilizes the event graph model to represent a program run. By connecting a set of specialized DeWiz modules (analysis, visualization etc.), the user can build an event graph processing pipeline. The different modules communicate using TCP/IP which makes it possible to distribute

a DeWiz system across several computers. This loose coupling of the modules contributes to the flexibility of DeWiz, but causes an administrative overhead and performance problems during on-line monitoring and analysis activities.

3 Our Approach

We consider PARADIS a tool for the breakdown of distributed programs, like on-line database systems with plenty of users, eBusiness, eCommerce or eGovernment systems, to name only a few. Nevertheless, the techniques might also prove very useful in the field of high performance computing, where message passing and shared memory computing is common. Our intent is to allow users to define events (eg. send and receive events when using MPI [6]), which are logged during a program run. These events form an event graph which provides the basis for our investigations. An event graph [7] is a directed graph, where the nodes are events and the edges stand for relations between the events.

To obtain a description of the application flow it is necessary to order the events in a causal manner. For this purpose we apply Lamport's [8] "happened-before relation". While the order for events occurring on one given object (node, process, thread,...) is given implicitly by the real-time timestamps of the events on the calling object with a given local time, the creation of relations between two dependent events on different objects can be more complicated if considering distributed systems without any global clock and where the local clocks are not synchronized and drifting. To get these events ordered we use a logical time-stamping mechanism (totally ordered logical clocks [9]).

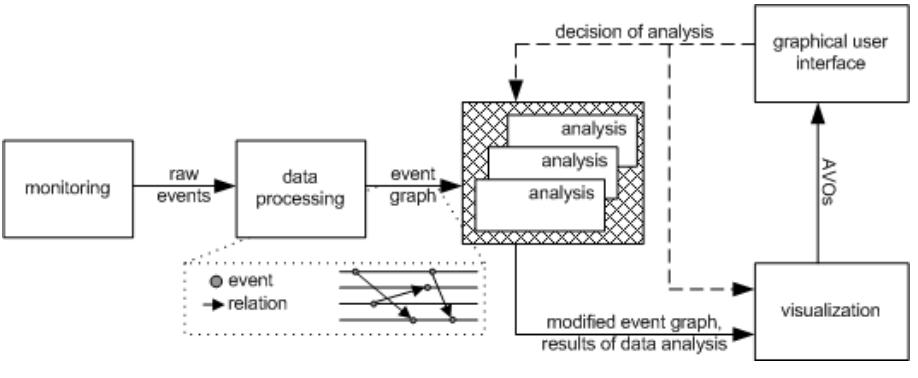


Fig. 1. Block diagram of the PARADIS system. An event graph is constructed from the recorded trace data and represents the basis for all further analysis tasks

Based on these conditions we create the event graph and offer various modules for textual and graphical representations and examinations. Figure 1 shows the logical units of PARADIS and their communication and is explained in the following sections.

3.1 Monitoring

To enable event tracing, first the program has to be instrumented. At the moment this is done statically by inserting calls to dedicated monitoring functions at points of interest in the source code of the inspected program. Being aware of the limitations due to the need of recompilation we are working on a dynamic instrumentation module using dyninstAPI [10]. It will allow the examination of already running programs without having to change the source code.

After the instrumentation each participating node in the computing environment executes a modified program which logs program state information. To comply with our system, for each event the values described in Table 1 (*Tr*) have to be recorded. Each event belongs to a particular category denoted by *type*. Possible categories are "communication", "critical section", "function call", "inspection of variable", and also user defined ones. Each *type* can have several subtypes. *Identification* is usually the id of the thread and the name or address of the node where the event has occurred. *Timestamp* is used to calculate durations (eg. blocking times) and statistics. For the creation of happened-before relations we utilize the logical Lamport time which is stored in the field *logicalclock*.

Table 1. Information stored for each event in the PARADIS monitoring environment. *Kind* indicates the traced (*Tr*) and the deduced (*De*) information

Information	Description	Kind
type	Category to which the event belongs	Tr
identification	Id of the thread and machine that produced the event	Tr
timestamp	Local physical time measured at the occurence of the event	Tr
logicalclock	Used to reconstruct the logical sequence in the event graph	Tr
data	Any other information for the later analysis	Tr/De
isHeadOf	Events that relate as source events to this event	De
isTailOf	Events to which this event does relate as a source event	De

The field *data* serves as storage for any further event-specific information. For instance to find all events referring to the same critical section one has to store the id of the critical section for which the events occurred. Since we use PARADIS for the analysis of distributed transaction-based applications we also store the id of the triggering transaction for each event. This information offers the possibility to trace the processing of a complete transaction in a distributed environment and to locate dependencies between single transactions.

3.2 Data Processing

The data processing unit commutes the raw events recorded during a monitored program run into the internal data model where it is negligible whether the analysis takes place post-mortem or on-line. For holding the relations, PARADIS offers two data structures (see Table 1) for each event. These two structures,

isHeadOf and *isTailOf*, store the connected events according to the event graph definition and the application specific relations.

Figure 2 shows a more detailed image of the data processing unit. Each component of the program which takes part in the monitoring process is recording program information according to its instrumentation. These "raw events" are gathered by Trace Collectors (TC). It is necessary to offer different techniques for the collection to accommodate different node types. For instance nodes in a cluster with a shared file system may store their traces in dedicated directories from where the TC fetches them, while single clients which do not grant access to a TC will most probably prefer sending their traces in uncertain time intervals. PARADIS allows new nodes to start, and already participating nodes to discontinue partaking in the monitoring process at any time.

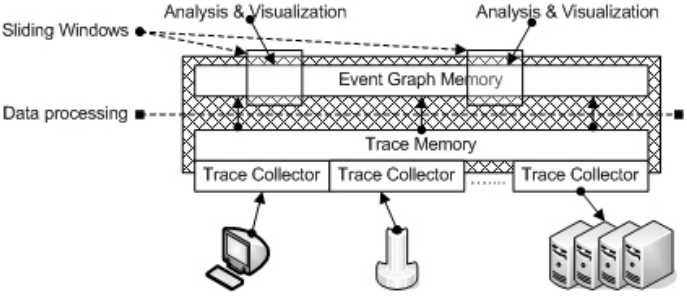


Fig. 2. Trace collecting and processing mechanism of PARADIS

Each raw event gets stored to the Event Graph Memory which offers a fast and efficient indexing and caching mechanism for the ongoing creation of relations, graph transformations and examinations. If more raw events are produced than the data processing unit can handle at a certain time, we use a selection mechanism, named "wave-front analysis", where those events are worked up first which occurred in neighbouring time segments. This leads to more significant results, as the event graph gets more detailed in the respective time frames and it is more likely to find related events. Additionally it is possible to filter specific events, for instance only those which cause blocking, to get intrinsic information even during high workloads.

3.3 Analysis

This unit works directly on the event graph which leads to a low main memory usage as there exists at most one single instance of any event at any time. The event graph memory module caches clusters of the most likely to be requested events ("sliding window"). By design an unlimited number of analysis modules may be active simultaneously while the data processing unit is still generating the graph. At the moment the analysis unit implements several filtering techniques, like for example by time of appearance or by blocking type to name only a few,

but it is possible to extend the set with graph transformation, pattern matching or export functionality for transferring PARADIS data to other graph analysis or visualization tools like VisWiz [11] or GraphML [12].

As PARADIS additionally provides support for events which occurred in the context of transactions, it governs the identifications of all transactions, to accelerate access to the associated events. This is necessary because multiple nodes and processes may work on the same transaction in parallel which can lead to a spreading of events which belong semantically together in the event graph memory.

3.4 Visualization

For simply visualizing communication between a handful of processes a space-time diagram like the one adumbrated in Figure 1 may be sufficient, but for large distributed programs with a vast number of processes, events and relations it may not. As our target programs are transaction-based applications we propose an abstraction on transaction level. Figure 3a shows an overview of some transactions of an example application, where a single bar represents one single transaction. Dedicated events like the blocking time before obtaining a critical section or establishing communication are emphasized on the lower and upper half of the bar and with different colors. Thereby one gets an impression on how long each transaction took place and how much time was spent unproductive due to blocking. As a bar contains all events of one transaction regardless on which object they occurred it is necessary to provide a more detailed view for the analysis of the communication patterns between the different nodes and processes in the distributed environment (see Figure 3b).

To ease the detection of reasons of an unintended runtime behavior, PARADIS supports the user by visually emphasizing different events and blocking times. For figuring out the reasons for race conditions or communication patterns, transactions that relate directly (communication partners, previous owner of critical sections,...) to the transaction under analysis are displayed too. The abstraction from transaction layer to process layer and the detailed information about each single event contribute to the systematic breakdown of distributed applications. To maintain independence from the user interface and the output device the visualization unit creates AVOs [13] (Abstract Visualization Object). The graphical representation of these objects is chosen by the viewing component. At the moment our graphical user interface is running under Microsoft Windows and the AVOs are rendered with OpenGL.

4 Results

We have tested our tool with traces of a real-world document-flow and eBusiness system. Users submit tasks using a web interface and the system executes the tasks by distributing subtasks to different layers (client layer, frontend server, backend server). In order to fulfil one task at least one transaction is created. The

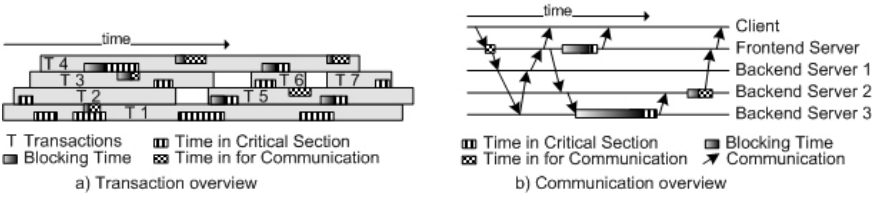


Fig. 3. The Transaction overview (a) shows all transactions in a given period, while the Communication overview (b) represents the inter-node communication of a single transaction

application was instrumented to produce events which comprehend the necessary information for the ensuing analysis. It was then started on a test environment, which is able to simulate real-world conditions. As this application is based on transactions, each recorded event can be associated with a specific transaction. One program run generates in one minute about 3000 transactions, 5.6 million critical section operations (request, enter, leave) and 76000 delegation operations (http and rpc requests, execution requests,...) which leads to a total of almost 6 million events per minute.

We have implemented several analysis and visualization modules like the one described in the previous section and especially the transaction overview proved very valuable for getting a first impression which transactions were unproductive over longer periods. With this information a goal-oriented search for the causes was possible, though we must admit that for really large numbers of transactions merging and filtering techniques become necessary to hide complexity.

5 Future Work

According to our tests the close coupling of the data processing and analysis units contributes causally to the celerity of PARADIS. Future work focuses on two aspects: the first is tuning our tool. We are optimizing the most time consuming tasks and want to reduce the amount of events, without losing important program information. One approach is the fractional outsourcing of the generation of the event graph, where relations between events which do not represent any type of inter-node communication are created on the nodes where they occurred. Another technique is the introduction of "meta-events" which encapsulate more than one event. This reduces the number of events and lessens the access to the event graph memory, but needs more effort in administration.

The second aspect is how to widen the field of application. We are developing new visualizations like highlighting positions with surpassing blocking times. Furthermore we are designing a mechanism for setting checkpoints in order to use record and replay techniques to repeat applications with non-deterministic behavior for debugging purpose.

Acknowledgements

This work was funded by ARC Seibersdorf research GmbH Austria and was originally initiated by an Austrian software company, which develops large distributed applications. The Institute of Graphics and Parallel Processing, at the Joh. Kepler University Linz/Austria was taking part as a consulting member.

References

1. B. P. Miller, M. D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K. L. Karavanic, K. Kunchithapadam, T. Newhall. The Paradyn Parallel Performance Measurement Tools. IEEE Computer 28, pp. 37-46, November 1995.
2. H. Brunst, H.-Ch. Hoppe, W. E. Nagel, M. Winkler. Performance Optimization for Large Scale Computing: The Scalable VAMPIR Approach. In V. N. Alexandrov, J. Dongarra, B. A. Juliano, R. S. Renner, C. J. K. Tan (Eds.): International Conference on Computational Science (ICCS), San Francisco, CA, USA, May 28-30, 2001, Proceedings, Part II. , Springer, LNCS 2074, pp. 751-760, 2001.
3. H. Brunst, W. E. Nagel, Allen D. Malony. A Distributed Performance Analysis Architecture for Clusters. IEEE International Conference on Cluster Computing, Cluster 2003, IEEE Computer Society, Hong Kong, China, pp. 73-81, December 2003.
4. S. Moore, D. Cronk, K. London, J. Dongarra. Review of Performance Analysis Tools for MPI Parallel Programs. In Proceedings of the 8th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springer, pp. 241-248, 2001.
5. C. Schaubschläger, D. Kranzlmüller, J. Volkert. Event-based Program Analysis with DeWiz. In M. Ronsse, K. De Bosschere (Eds.): Proceedings of the Fifth International Workshop on Automated Debugging (AADEBUG 2003), Ghent, September 2003.
6. Message Passing Interface Forum, MPI: A Message-Passing Interface Standard - Version 1.1, `http://www.mcs.anl.gov/mpi/`, June 1995.
7. D. Kranzlmüller. Event graph analysis for debugging massively parallel programs. PhD Thesis, Institute for Graphics and Parallel Processing, Joh. Kepler University Linz, Austria, `http://www.gup.uni-linz.ac.at/~dk/thesis/`, September 2000.
8. L. Lamport. Time, clocks, and the ordering of events in a distributed system. In Communications of the ACM, Vol. 21, No. 7, pp. 558-565, July 1978.
9. C. Fidge. Fundamentals of Distributed System Observation. In IEEE Software, Volume 13, pp. 77-83, 1996.
10. B. Buck, J. Hollingsworth. An API for Runtime Code Patching. In Journal of High Performance Computing Applications, 14(4), pp. 317-329, 2000.
11. R. Kobler, Ch. Schaubschläger, B. Aichinger, D. Kranzlmüller, J. Volkert. Examples of Monitoring and Program Analysis Activities with DeWiz. In Proc. DAPSYS 2004, pp. 73-81, Budapest, Hungary, September 2004.
12. U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, M. S. Marshall. GraphML Progress Report: Structural Layer, Proposal.Proc. 9th Intl. Symp. Graph Drawing (GD '01), LNCS 2265, pp. 501-512, 2001.
13. R. B. Haber, D. A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In G. Nielson, B. Shriver, L. J. Rosenblum: Visualization in Scientific Computing, pp. 74-93, IEEE Comp. Society Press, 1990.

Automatic Tuning of Data Distribution Using Factoring in Master/Worker Applications[1]

Anna Morajko, Paola Caymes, Tomàs Margalef, and Emilio Luque

Computer Science Department. Universitat Autònoma de Barcelona,
08193 Bellaterra, Spain
{ania, paola}@aomail.uab.es
{tomas.margalef, emilio.luque}@uab.es

Abstract. Parallel/Distributed programming is a complex task that requires a high degree of expertise to fulfill the expectations of high performance computation. On the one hand, application developers must tackle new programming paradigms, languages, libraries. On the other hand they must consider all the issues concerning application performance. On this context the Master/Worker paradigm appears as one of the most commonly used because it is quite easy to understand and there is a wide range of applications that match this paradigm. However, to reach high performance indeces it is necessary to tune the data distribution or the number of Workers considering the particular features of each run or even the actual behavior that can change dynamically during the execution. Dynamic tuning becomes a necessary and promising approach to reach the desired indeces. In this paper, we show the usage of a dynamic tuning environment that allows for adapting the data distribution applying Factoring algorithm on Master/Worker applications. The results show that such approach improves the execution time significantly when the application modifies its behavior during its execution.

1 Introduction

Parallel and distributed systems offer high computing capabilities that are used in many scientific research fields. These capabilities have taken the evolution of science to a new step called computational science. This facilitated the determining of the human genome, computing the atomic interactions in a molecule or simulating the evolution of the universe. So biologists, chemists, physicians and many other researchers have become intensive users of high performance computing. They submit very large applications to powerful systems in order to get the results as fast as possible, considering the largest problem size and taking advantage of the resources available in the system. The increasing need of high performance systems has driven scientists towards parallel/distributed systems. Although such systems have their performance limits, they are much more powerful than the rest of the computers and hence are better for solving scientific problems demanding intensive computation.

[1] This work has been supported by the MCyT (Spain) under contract TIC2001-2592 and has been partially supported by the *Generalitat de Catalunya* – GRC 2001SGR-00218.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 132 – 139, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Problems covered by computational science usually comprise a lot of calculations which are easily parallelizable, and consequently applications to solve these problems can be designed using the Master/Worker (M/W) paradigm [1]. Typically, in the M/W paradigm a Master process in each iteration distributes data among Worker processes and waits for their response. Each Worker processes its portion of data independently of the remaining Workers and returns the results to the Master. When the Master receives results from the Workers, it may distribute the data again. There are many cases the Master must synchronize all the results from all the Workers before the next data distribution. However, heterogeneous computing and communication powers, or varying the amount of distributed data could cause load imbalance. In these situations slower or overloaded machines and/or incorrect data distribution may significantly increase the idle time of processes and influence into the application execution time. The workload balancing goal therefore is to minimize the idle time and calculate the right amount of data for each process. Load balancing should be achieved in such a way that fast computers will automatically process more amount of data than the slower ones. Moreover, an optimal data distribution may also depend on dynamic factors such as input data, network load, computer load and so on. Before the application execution, developers do not know these parameters, hence they cannot distribute the work properly. Therefore, it can be beneficial to dynamically tune the data distribution by adapting it to changing conditions. In this context, our goal is to balance and distribute data correctly and dynamically among the available processes taking into account capabilities and load of the machines the application runs on.

In the following sections of this paper, we present a complete performance optimization scenario that considers the data distribution problem in a dynamic approach. In Section 2, we describe the algorithm used to distribute the data. In Section 3, we analyze the tuning of the data distribution by using the MATE environment that supports dynamic tuning of parallel applications. In Section 4, we present the results of experiments conducted in the MATE to dynamically tune the data distribution using the presented algorithm. Finally, Section 5 shows the conclusions of this work.

2 Factoring Data Distribution

The data distribution from a Master process to the Worker ones can be done in several different ways considering different algorithms. One possibility is to divide the total data n in p chunks, where p is the number of Workers and distribute one of these chunks to each Worker. However, if there is any heterogeneity in the system the faster processors will be waiting for the slower ones. Another possibility is to divide the total data in m same-size chunks, where m is greater than p. In this case, the Master distribute the p first chunks to the Workers. When one Worker finishes its work, it returns the result to the Master and if there are chunks to be processed, the Master sends to that Worker a new chunk. A third possibility is to create chunks of different size in such a way that the initial distribution consider bigger chunks and when there are less data to be processed, the amount distributed is smaller.

One of the algorithms using this third approach is the Factoring Scheduling algorithm [2]. In the factoring algorithm, the work is partitioned according to a factor into a set of different-size tuples. The Master distributes one tuple to each Worker. If a

Table 1. Examples of tuple sizes for different factors

Work size (N)	Number of Workers (P)	Factor (f)	Threshold (T)	Tuples
1000	2	1	1	500,500
1000	2	0.5	1	250,250,125,125,63,63,32,32,16,16,8,8,4,4, 2,2,1,1
1000	2	0.5	16	250, 250, 125, 125, 62, 62, 32, 32, 16, 16, 16, 16
1000	2	0.7	1	350, 350, 105, 105, 31, 31, 10, 10, 3, 3, 1, 1
1000	4	1	1	250, 250, 250, 250
1000	4	0.5	1	125, 125, 125, 125, 62, 62, 62, 62, 32, 32, 32, 32, 16, 16, 16, 16, 8, 8, 8, 8, 4, 4, 4, 4, 2, 2, 2, 2, 1, 1, 1, 1
1000	4	0.5	16	125, 125, 125, 125, 62, 62, 62, 62, 32, 32, 32, 32, 16, 16, 16, 16, 16, 16, 16, 16
1000	4	0.7	1	175, 175, 175, 175, 52, 52, 52, 52, 16, 16, 16, 16, 5, 5, 5, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1,

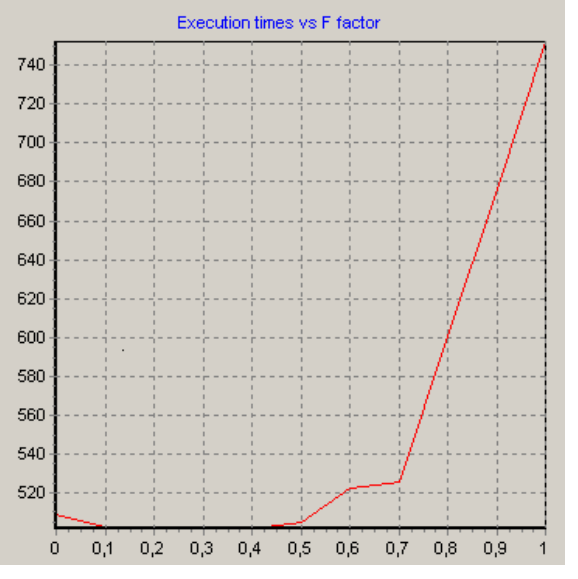


Fig. 1. Execution time in function of factor f (N=1000, P=4, T=1, relative speeds: 1,3,3,3)

Worker has finished its tuple and there are tuples to be processed, then it receives the next tuple. This cycle is repeated till all the tuples of the work are processed.

Assuming there are P parallel Workers, a threshold T>0 (minimal tuple size) and a factoring value (0<f<=1), the sizes of factored work tuples are calculated according to the following algorithm [2]:

```
R0 = N (total work size)
Repeat
        For each P
                Gi = Ri * f / P        //Gi: ith tuple size
        Ri+1 = Ri - (P*Gi)             //Ri+1: remaining work size
until Ri < T.
```

The example tuple sizes calculated according to the presented above factoring algorithm are shown in Table 1.

Figure 1 shows the execution time of an example application in a particular condition considering different values for the factor f. As it can be observed, the value of the factor f influences strongly the execution time of the application.

3 Tuning Factor f with MATE

To provide dynamic automatic tuning of parallel/distributed applications we have developed an environment called MATE (Monitoring, Analysis and Tuning Environment) [3]. MATE performs dynamic tuning in three basic and continuous phases: monitoring, performance analysis and modifications. This environment dynamically and automatically instruments a running application to gather information about the application behavior. The analysis phase receives events, searches for bottlenecks, detects their causes and gives solutions on how to overcome them. Finally, the application is dynamically tuned by applying given solution. Moreover, while it is being tuned, the application does not need to be re-compiled, re-linked and restarted.

The knowledge to represent the performance model of each particular performance problem is specified in a component called "*tunlet*". Each tunlet includes the information concerning the measure points to insert instrumentation in the target application, the performance model to determine the behavior of the application and the required modifications, and the tuning actions to improve the application performance.

We have been conducted many various practical experiments on parallel/distributed applications to see if our approach really works. We have proven that it is effective, profitable, and can be used for a real improvement of the program performance. Running applications under control of MATE has allowed for adapting their behavior to the existing conditions and improving their functionality [4].

To dynamically tune the data distribution we had to consider several issues:

- How to calculate work tuples (factoring algorithm described in previous section).
- How to distribute the data according to the given factor.
- How to calculate the optimal Factor and what is required to calculate this value.

To tune the Factor f applying the MATE environment, the application must be prepared for the tuning actions and hence it must implement adequately the data distribution:

```
For each iteration
        //according to the factoring algorithm
        Calculate work tuples for a given factor
        //first work distribution to all Workers
For each Worker
        Send corresponding tuple
        //if there are still tuples to be distributed
        While distributed tuples < Total number of tuples
        Receive results
        Send corresponding tuple
        //if there are still tuples to be completed
        While completed tuples < Total number of tuples
        Receive results
```

During the execution, we can monitor the PVM functions responsible for exchanging messages, e.g. pvm_send() and pvm_recv(). In particular, by monitoring: send entry/exit, receive entry/exit events in the Master process, and receive entry/exit and send entry/exit in all Worker processes, we are able to perform all necessary measurements. To find the optimal value of the Factor the performance model must provide the prediction of the iteration time. For this purpose we implemented the simulation of the application iteration time according to the following:

```
For each processor
     Calculate Latency
     Calculate Bandwidth
     Calculate work unit execution time (i.e. processor speed)
For each Factor f (0<f<=1)
     Calculate tuples
     Calculate application iteration execution time
     While there is work left
            For each process (using time counter)
                   If process is free
                                Assign tuple for process
                   Time++
Chose the minimal time of application iteration
Set Factor value to the minimal Factor value
```

The presented algorithm simulates the execution of the program iteration for each chosen possible value of Factor f. The simulation uses basic measurements such as network latency, bandwidth and current speed for each processor. The algorithm simulates the complete iteration by assigning subsequent work tuples to next free processor. It takes into account the time for sending the work, time for processing the work on the selected processor in function of its current speed (i.e. capacity and current load), and time for returning the result. The iteration finishes when all tuples have been processed, and the simulation records its execution time. After simulating iterations for all chosen factors, the algorithm returns the value f for the iteration with the shortest execution time.

Considering the algorithm presented above we must receive a set of metrics to calculate the optimal Factor such as, network latency, network bandwidth, average Worker speed expressed as the work unit processing time. To simplify the implementation, we have assumed network latency to be 1ms. Network bandwidth can be computed using the following formulas:

```
Tcomm = Tlatency + n * Tbandwith
Tbandwith = (Tcomm – Tlatency) / n
Tcomm = receive exit time on Master - send entry time on
Master – tuple processing time on a Worker
```

Work unit processing time is computed as the averaged value of the tuple processing time divided by number of work units per tuple. This is calculated individually for each Worker. The average is taken for all work units processed during last iteration. This assumes the total iteration time is not very large and the average can express current machine load. If the iteration time is long, this should be changed to an average taken for a given time window. Each iteration updates the work unit processing time value for each Worker and in this way we can keep track of dynamic variations in the load of the machines. It must be pointed out that this approach is suitable only if the work unit processing time depends on the work unit size and not on its content.

If a new value of the Factor is different than the current one, then the tuning action should be performed. The tuning action includes the modification of the value of specific predefined application variable. This variable represents the Factor of the work and its name is well known to the tunlet. When a next iteration is performed in the application, first, the new value of the Factor should be applied to recalculate the work tuples, and then the data can be distributed. The variable modification does not need to be synchronized, as the application will use its value next time the iteration of work processing starts.

4 Experimental Results

In this section, the experimental results obtained by applying the tuning environment to a real Master/Worker application are presented.

To conduct the experiments, we selected an intensive computing Forest Fire Propagation application called Xfire [5]. The Xfire application is a Master/Worker PVM based implementation of the simulation of the fireline propagation. It calculates the next position of the fireline considering the current fireline position and different aspects as weather, wind, vegetation, etc.

In every scenario, one Worker was executed in the same machine as Master. Remainder Workers were executed in individuals machines.

Experiments were conducted on a cluster of homogenous Pentium 4, 1.8 Ghz, (SuSE Linux 8.0) connected by 100Mb/sec network. Each experiment was performed three times and the average of the execution time for the application was calculated.

We have conducted our experiments in three scenarios:

- In the first scenario, Xfire was executed in dedicated machines without any tuning and it was used as a reference result.
- In the second scenario, Xfire was executed in a non-dedicated environment; in this case we have introduced a controlled variable load, using our Load Generator tool. We generated 75 % CPU load in a half of machines every eight iterations.

- The third scenario is exactly the same on the third one, but in this case Xfire was executed under MATE applying the tuning of the data distribution. In this way, we demanded more adaptability to the load balancing.

Table 2 summarizes the experimental results. These results are also presented in figure 2.

Table 2. Execution time of Xfire (in seconds) considering different number of Workers

Scenario \ Number of Workers	1	2	4	8	16	30
Xfire	17663	9126	4732	2488	1350	880
Xfire + variable load		11906	6098	3157	1501	1208
Xfire + MATE + variable load		10206	5684	2784	1487	1073

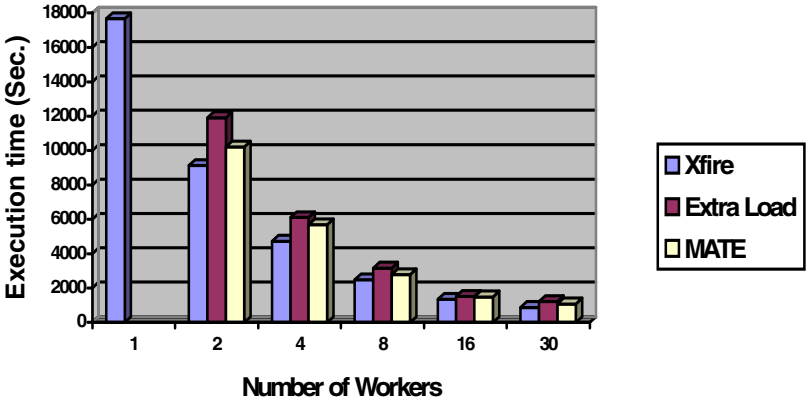


Fig. 2. Execution time of Xfire considering different number of Workers

This figure shows the execution time of the Xfire application considering different number of Workers, the execution time of Xfire with additional load and the execution time of Xfire with additional load under the control of MATE. It can be observed that the introduction of an extra load provokes an increase in the execution time of the application. This is due to the new extra load, but also to the load imbalance introduced by the wrong Factor distribution in the new situation. The execution under MATE control is able to detect the load imbalance and correct Factor to improve the execution time.

5 Conclusions

Parallel and distributed programming offers high computing capabilities to the users in many scientific research fields. The performance of applications written for such environments is one of the crucial issues. Data distribution appears as one of the most significant problems in Master/Worker applications. There are several approaches to

distribute data, but the main goal is to accomplish that there are no Workers idle waiting for busy ones. One of this approaches is the Factoring Scheduling. However, the factoring requires the tuning of the Factor f that can change dynamically depending on system load. In this context we show the applicability of the MATE dynamic tuning environment to improve the data distribution and the application performance by tuning the value of the Factor f. The experimental results show that the MATE environment is very useful and improves the execution time significantly adapting application to changing conditions.

References

1. César, E., Mesa, J.G., Sorribes, J., Luque, E. "Modeling Master-Worker Applications in POETRIES". IEEE 9th International Workshop HIPS 2004, IPDPS, pp. 22-30. April, 2004.
2. Hummel, S.F., Schonberg, E., Flynn, L.E., "Factoring: A method for Scheduling Parallel Loops", Communications of the ACM, 35, no 8, August 1992.
3. Morajko, A., Morajko, O., Jorba, J., Margalef, T., Luque, E. "Dynamic Performance Tuning of Distributed Programming Libraries". LNCS, 2660, pp. 191-200. 2003.
4. Morajko, A., Morajko, O., Margalef, T., Luque, E.. "MATE: Dynamic Performance Tuning Environment". LNCS, 3149, pp. 98-107 (2004).
5. Jorba, J., Margalef, T., Luque, E., Andre, J, Viegas, D.X. "Application of Parallel Computing to the Simulation of Forest Fire Propagation", Proc. 3rd International Conference in Forest Fire Propagation, Vol. 1, pp. 891-900. Portugal, November 1998.

DynTG: A Tool for Interactive, Dynamic Instrumentation*

Martin Schulz, John May, and John Gyllenhaal

Center for Applied Scientific Computing,
Lawrence Livermore National Laboratory
{schulzm, johnmay, gyllen}@llnl.gov

Abstract. With the increasing complexity of today's systems, detailed performance analysis is more important than ever. We have developed DynTG, a tool for interactive, dynamic instrumentation. It uses performance module plugins to reconfigure the data acquisition and provides a source browser that allows users to insert any probe functionality provided by the modules dynamically into the target application. Any instrumentation can be added both before and during the application's execution and the acquired data is presented in realtime within the source viewer. This enables users to monitor their applications' progress and interactively control and adapt the instrumentation based on their observations.

1 Motivation

Users require comprehensive performance toolkits to analyze their applications, find bottlenecks, and derive appropriate optimizations. Many research projects have focused on this problem and have developed a large range of such tools. Some of the more prominent examples are TAU/ParaProf [1], SvPablo [4], and Kojak [9]. Also several commercial tools are available, including Intel's Vtune and SGI's SpeedShop. All of them allow users to gather a large variety of performance metrics and often provide graphical user interfaces to present the results. However, most of them rely on static source code instrumentation and/or post-mortem performance analysis and hence lack interactive capabilities.

Dynamic instrumentation is the first step towards interactive tools. Most tools in this area are built on top of DynInst [2], an API to dynamically insert arbitrary code snippets into running applications. Performance monitoring infrastructures, like OMIS [6] or DPCL [3], and tools like Paradyn [8] use this mechanism to insert performance probes into the application. However, these tools don't provide the user with interactive, source-code based environments to control their instrumentation.

* This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. UCRL-CONF-209840.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 140–148, 2005.
© Springer-Verlag Berlin Heidelberg 2005

With DynTG, we have developed a tool that integrates dynamic instrumentation with a source browser and provides users with a fully interactive performance analysis environment. It builds on top of DPCL to dynamically instrument applications and to acquire realtime performance data. The actual performance probes are dynamically loaded into DynTG using a modular plugin concept. By providing new performance modules, users can reconfigure DynTG to suite their particular needs and extend it with new performance metrics and data sources.

All instrumentation is controlled through a scalable source browser developed using the Tool Gear framework [7]. Performance modules dynamically register their data sources and instrumentation actions with the Tool Gear based GUI and thereby dynamically customize the tool based on their requirements. Further, all performance data gathered by the probe modules is transmitted to the GUI and displayed inside the source browser in realtime. This allows users to observe changes in the application behavior, detect bottlenecks, and, if necessary, add or remove instrumentation at runtime.

2 The DynTG Architecture

A tool for interactive, dynamic instrumentation must a) provide an extensive and extensible set of data probes; and b) allow users to insert instrumentation into their codes through a source browser. DynTG achieves these two goals by a) providing a plugin concept using dynamically loaded performance modules; and b) relying on and extending Tool Gear [7], a versatile framework for building graphical tools. DynTG enables users to specify instrumentation locations directly in the source code and to select appropriate data probes at these locations. Once selected, DynTG uses DPCL [3]/DynInst [2] to instrument the target application with the selected probe functionality. The resulting performance data is returned to the GUI and displayed within the source browser on a per-line basis.

Figure 1 shows the basic architecture of DynTG. It is divided into three major components — performance probes, the collector, and the GUI. The GUI

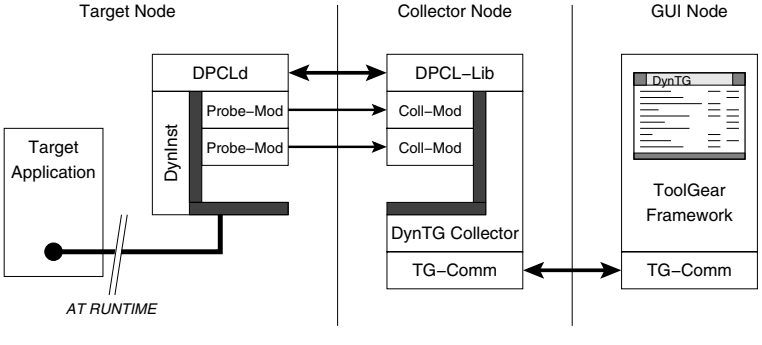


Fig. 1. Architecture of DynTG

allows the user to spawn and control the collector, which acts as a link or proxy between the application and the GUI. Once started, it uses DPCL to launch and control the application on the target system and to install the probe modules. These modules return any acquired performance data through DPCL to corresponding collector modules. There, the performance data is preprocessed and then forwarded through the collector infrastructure to the GUI for display using the Tool Gear communication mechanism.

Since both libraries use standard sockets for the actual communication, we can (if necessary) place each component onto a different node. For example, the GUI can run on the user's desktop and communicate with a remote collector on a front-end node, which then interacts with the probe modules on the application's compute nodes. This matches the structure of typical cluster or computer center configurations and has the ability to provide a fast, interactive GUI, while enabling online accesses to running applications on remote systems.

3 Probe/Collector Modules

DynTG has the ability to dynamically load and activate performance modules. This allows users to extend, configure, and customize the tool according the specific monitoring needs.

3.1 Module Concept

Each module is implemented in two separate parts: a probe module, which is dynamically inserted into the running application; and a collector module, which is loaded into the DynTG collector.

Probe modules are generally written in C and implement custom data probes. They communicate the acquired results back to the corresponding collector module using DPCL's transfer mechanisms. In most cases this transfer is accomplished with a single call to DPCL's *Ais_send*.

The corresponding collector module implements the necessary DPCL callback, which receives the data, if necessary transforms it into an appropriate format, potentially performs a preliminary analysis, and then hands the data to the GUI. In addition, the collector module implements two management routines required by the framework: a query routine, which returns a list of requirements for this modules along with a short textual description of its functionality, and an initialization routine.

The module interfaces are kept small by moving common and low-level functionality into the framework or the communication libraries. This simplifies the module development process, lowers the learning curve, and hence allows even unexperienced users not familiar with DPCL or Tool Gear to develop new probe/collector modules in a short amount of time and thereby to dynamically extend DynTG without changing the source of either collector or GUI.

3.2 Installation and Activation

During startup, the collector scans a module library and queries each module found for its installation requirements using the query routine introduced above. Based on the results it excludes incompatible modules (e.g., those that rely on specialized hardware or software) and then loads and activates the remaining ones. During this activation, the module locates the corresponding probe module and links it to a DPCL callback.

After all probes are loaded and initiated, the collector registers the module's functionality with Tool Gear: it compiles a list of possible actions (e.g., potential instrumentation insertion and removal events) and data sources, and then sends this list to the GUI. Together with this registration, it establishes a name space, which is later used to react on GUI actions or send observed data events.

3.3 Module Selection

It is not always necessary or beneficial to load and activate all available instrumentation modules. Each module consumes resources and also might incur conflicts with other modules. This is especially critical with an increasing number of modules. DynTG therefore enables the user to select and activate a subset of all available modules. This can either be done using a command line parameter during the collector startup or interactively through the DynTG GUI. For the latter, the collector forwards a list of all available modules together with the short description returned by the above mentioned query routine to the GUI. The GUI then builds a selection dialog (see Figure 2/right,bottom) and queries the user for input. Based on the user's decision, the GUI requests the activation of the selected modules in the DynTG collector.

3.4 Existing Modules

Currently, we have implemented three probe/collector modules:

Counter: The counter module provides a counter that counts how often a program executes a specific location. This module can be used to examine iteration counts, distribution of execution paths on conditionals, or callgraph coverage.

Timer: The timer module provides wall clock timing using the UNIX *gettimeofday* call. The timer is controlled by two actions that the user can instrument the code with: start timer and stop timer. A stop timer action thereby always corresponds to the most recently started active timer and computes the time difference between the two. In order to keep track of these correspondences, the timer module maintains a timer stack: during a start timer action, the current time value is pushed onto the stack and during a stop, this value is retrieved from the stack and used to compute the time difference.

PMAPI: This module provides direct access the hardware counters on IBM Power systems using AIX's Performance Monitoring API (PMAPI). Due to hardware constraints only a subset of events can be accessed at the same time.

Table 1. Implementation complexity of probe/collector modules in lines of code

Module	Probe	Collector
Counter	10	157
Timer	170	171
PMAPI (+ shared utility)	99 (+199)	213 (+199)

To further complicate things, each of the eight hardware counters available in Power-3 or Power-4 systems can only be used for a specific and distinctly different set of events. During the loadtime of the PMAPI collector module, the user specifies a set of events of interest in a priority list[1]. The module then computes an optimal mapping of this list to the available hardware resources on the target architecture and configures the hardware counters appropriately. Once completed, the PMAPI module registers its start and stop events for each available and configured counter and maintains a stack, similar to the one in the timer module, to associate the corresponding start and stop events.

Table 1 shows the number of lines used for the implementation of each module (including comments). Despite the fact that lines of codes are naturally a very inaccurate measurement, the numbers show that the implementation complexity for each module is low. The collector modules show a constant overhead needed to load and register the module, while the complexity in the probe module only depends on the complexity of the actual probe functionality.

4 GUI Support for Interactive Instrumentation

Dynamic instrumentation enables users to selectively activate instrumentation in their codes. In order to fully take advantage of this feature, users must be able to interactively browse their source code, identify possible instrumentation locations, and interactively choose the required instrumentation at those points. DynTG uses Tool Gear [7] to implement this functionality and extends it to provide reconfigurable instrumentation and support for modules.

4.1 Integrating DynTG with the Tool Gear Framework

Source code viewers are complex and difficult to implement, yet required by a large number of tools. The Tool Gear project [7] was born based on this observation and provides a versatile framework for the development of interactive tools. As its core it provides a scalable source viewer, which can display both the source code itself as well as multiple performance metrics on a per source line basis. The performance data itself is stored inside Tool Gear using a database.

[1] Currently this list is specified in a configuration file; we are currently working on a GUI version of this selection process

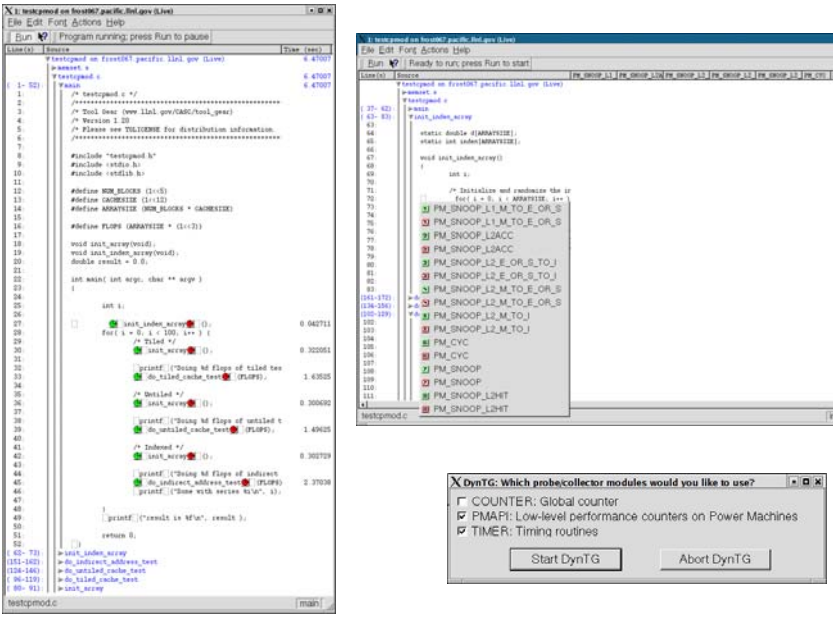


Fig. 2. DynTG GUI: source browser with timer experiment (left), context-sensitive action menu for PMAPI module (right,top), module selection dialog (right,bottom)

Data can directly be added to this database from a collector using the Tool Gear communication library.

The Tool Gear source browser provides a collapsible view of all source modules on the left side of the window and displays the corresponding performance information in column style for each source line on the right of it. We use this generic setup to display performance data acquired by the probe modules. Each module can register one or more data sources and each of these sources will be displayed using a separate column within the source browser. The DynTG collector continuously updates these performance observations during the program's runtime. This enables the user to monitor the progress of their code, spot potential bottlenecks early, and react by inserting additional or removing unnecessary instrumentation. In addition, Tool Gear aggregates the presented performance data hierarchically showing the performance of a complete function, module, or application.

Figure 2 (left) shows a typical view of the source code browser as it is used by DynTG. In this example we use the timer probe/collector module to instrument a simple test application and to measure wall clock times of selected function invocations within *main*. We place start timer instrumentations (green icons) before and end time instrumentations (red icons) after the relevant function calls. The results are displayed in a single data column on the right and are updated every time the application executes one of the instrumentation points. In addition, the top rows show the accumulated runtime of all counters in the main routine, the *testcpmod.c* module, and the total application.

4.2 DynTG Startup and Configuration

During the startup of DynTG, the Tool Gear infrastructure launches the DynTG collector on the remote node and establishes the communication using its communication library. As described in Section 3.2, the collector then searches for all available modules and activates a user-selected subset of those modules. During their activation, the module compiles a list of actions provided by this module and forwards this to the GUI. Examples for this can be "Start a timer" or "Count at this location". When the GUI receives a request for a new action, it selects an appropriate icon and adds it to the list of available actions.

This runtime configuration process guarantees that the GUI is independent from the collector: the GUI itself does not enable or provide any functionality to the user. Instead, all actions required by a collector are defined and installed by the collector. As a consequence, new modules can easily be added without modifying the GUI code itself. The only exception are probes that require new datatypes for the performance database or new visualization or display modes. Most performance modules, however, are able to work with the existing infrastructure.

In addition, the GUI queries possible instrumentation points from the collector at startup and marks them in the source display. At these locations DynTG offers a context-sensitive instrumentation menu that shows all possible instrumentation actions that can be inserted or removed at that particular location. The user can then select from this menu and insert the instrumentation. In addition, the DynTG includes a bulk instrumentation mechanism to instrument a set of routines. For this features, the user selects the instrumentation action as above and specifies the name of the routines to be instrumentation using regular expressions.

Any instrumentation can be done both before the program's execution (in which case the performance data from the whole run is captured) or at runtime (in which case the performance data is gathered from the time of the instrumentation). It is further possible to dynamically remove instrumentation points.

Figure 2 (right,top) shows an example of such an instrumentation menu. In this example, the PMAPI module is loaded and instantiated with eight different native hardware counters. Consequently the menu has 16 entries: a start (green icons) and a stop (red icons) event for each counter type. Further, the GUI displays eight columns for the performance data, again one for each hardware counter. In both cases, the collector has queried the counter names from PMAPI and registered them with the GUI dynamically.

5 Experiments

In the following section we present experimental results quantifying the overhead caused by DynTG. All experiments were conducted on one CPU of a dedicated 16-way IBM Power-3 system. We use two numerical applications from the ASC Purple Benchmark Codes [5]: SMG2000 with a working set of 60x60x60 and sPPM with a working set of 64x64x64. We inserted a single instrumentation

Table 2. Runtime and Overhead measurements of the three existing DynTG modules

	Absolute runtime				Overhead		
Benchmark	Baseline	Counter	Timer	PMAPI	Counter	Timer	PMAPI
SMG2000	56.6s	57.7s	58.0s	58.0s	1.94%	2.47%	2.47%
sPPM	53.4s	56.7s	59.5s	60.0s	6.18%	11.4%	12.4%

action into a central location and measured the overhead. This location was executed 34167 times in SMG2000 and 245760 times in sPPM.

As the results in Table 2 show, DynTG has an low impact on performance even if the instrumentation is executed very frequently like in sPPM (over 4000 times per second). In general, the overhead corresponds to the work performed within the data probes inserted into the application. Consequently, the overhead of the *Timer* and *PMAPI* modules is higher, since these modules require system calls to gather their data. In addition, we compared the runtime of an application run under the control of DynTG without instrumentation to the baseline and observed no difference in execution time.

6 Conclusions

In this paper, we presented DynTG, a tool for interactive, dynamic instrumentation. It includes a scalable source browser, in which the user can view all possible instrumentation points and insert probes into the application. Once the probes are installed, the performance data gathered by them is displayed in realtime inside the source browser at the location of the instrumentation point.

DynTG uses DPCL to realize dynamic instrumentation and relies on and extends Tool Gear to provide the scalable source browser. To connect those two components DynTG implements a reconfigurable collector that acts as a proxy between the data probes and the GUI. The probes as well as the probe specific parts of the collector are implemented as dynamically loadable plugin modules. This concept enables users to customize and reconfigure the tool depending on their actual monitoring requirements.

References

1. R. Bell, A. Malony, and S. Shende. A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis. In *Proceedings of the International Conference on Parallel and Distributed Computing (Euro-Par 2003)*, pages 17–26, Aug. 2003.
2. B. Buck and J. Hollingsworth. An API for runtime code patching. *The International Journal of High Performance Computing Applications*, 14(4):317–329, 2000.
3. L. DeRose, T. Hoover, and J. Hollingsworth. The Dynamic Probe Class Library — An Infrastructure for Developing Instrumentation for Performance Tools. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, Apr. 2001.

4. L. DeRose and D. Reed. SvPablo: A Multi-Language Architecture-Independent Performance Analysis System. In *Proceedings of the International Conference on Parallel Processing (ICPP)*, Sept. 1999.
5. Lawrence Livermore National Laboratory. The ASCI purple benchmark codes. http://www.llnl.gov/asci/purple/benchmarks/limited/code_list.html, Oct. 2002.
6. T. Ludwig, R. Wismüller, V. Sunderam, and A. Bode. *OMIS — On-line Monitoring Interface Specification (Version 2.0)*, volume 9 of *LRR-TUM Research Report Series*. Shaker Verlag, Aachen, Germany, 1997. ISBN 3-8265-3035-7.
7. J. May and J. Gyllenhaal. Tool Gear: Infrastructure for Parallel Tools. In *Proceedings of the 2003 International Conference on Parallel and Distributed Techniques and Applications*, June 2003.
8. B. Miller, M. Callaghan, J. Cargille, J. Hollingsworth, R. Irvin, K. Karavanic, K. Kunchithapadam, and T. Newhall. The Paradyn Parallel Performance Measurement Tool. *IEEE Computer*, 28(11):37–46, Nov. 1995.
9. B. Mohr and F. Wolf. KOJAK - A Tool Set for Automatic Performance Analysis of Parallel Programs. In *Proceedings of the International Conference on Parallel and Distributed Computing (Euro-Par 2003)*, pages 1301–1304, Aug. 2003.

Rapid Development of Application-Specific Network Performance Tests

Scott Pakin

lOS Alamos National Laboratory, Los Alamos, NM 87545, USA
pakin@lanl.gov
http://www.c3.lanl.gov/~pakin

Abstract. Analyzing the performance of networks and messaging layers is important for diagnosing anomalous performance in parallel applications. However, general-purpose benchmarks rarely provide sufficient insight into any particular application's behavior. What is needed is a facility for rapidly developing customized network performance tests that mimic an application's use of the network but allow for easier experimentation to help determine performance bottlenecks.

In this paper, we contrast four approaches to developing customized network performance tests: straight C, C with a helper library, Python with a helper library, and a domain-specific language. We show that while a special-purpose library can result in significant improvements in functionality without sacrificing language familiarity, the key to facilitating rapid development of network performances tests is to use a domain-specific language designed expressly for that purpose.

1 Introduction

Parallel applications utilize the interconnection network in a variety of ways, including nearest-neighbor communication on a 2-D or 3-D mesh/torus (e.g., in ocean-modeling codes [1]); hierarchical communication (e.g., in molecular-dynamics codes [2]); and, master/slave communication (e.g., in Monte Carlo codes [3]). However, general-purpose network performance tests such as NetPIPE [4], Mpptest [5], and those that appear in the Pallas MPI Benchmarks [6] and SKaMPI [7] suites, measure performance independently of any particular application's usage of the network. For example, it is common to measure network bandwidth as the peak data rate achieved when sending a large number of messages back-to-back between two otherwise idle endpoints, even though few applications utilize such a communication pattern. General-purpose tests are nevertheless important to application developers because they indicate – in a standard format – upper bounds in network performance that developers can use to determine if application performance is being limited by the network.

Special-purpose benchmarks targeted to a particular inquiry are an important complement to general-purpose benchmarks. For example, if an application runs significantly slower than a general-purpose test would indicate, it may be

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 149–157, 2005.
© Springer-Verlag Berlin Heidelberg 2005

worthwhile to extract the application's particular communication pattern into a separate test program and perform *in vivo* experiments with that (simulating message aggregation, varying message-buffer alignment, reducing communication granularity – all on a real cluster with a real network), the idea being that it is quicker and easier to modify a small test program than a large application. Unfortunately, special-purpose tests receive little attention in practice and in the literature because they can be time-consuming to write and debug; and, because they may be run only a few times before being discarded, few application developers consider the benefits worth the effort.

In this paper, we investigate three approaches intended to facilitate the rapid generation of special-purpose network performance tests and compare these to a baseline test written in standalone C. Section 2 describes the sample performance test which is used as the basis for comparison, lists the metrics used to evaluate the alternatives, and presents the baseline C implementation. Section 3 describes a library for performance testing and examines the improvement over the baseline C code when used with C and with Python. Then, Sect. 4 introduces the coNCePTuaL language, shows how the sample performance test can be rewritten in coNCePTuaL, and highlights the benefits of doing so. Finally, Sect. 5 draws some conclusions about the results of this study.

2 Problem Specification

When selecting a sample problem to use as a running example throughout this paper, the challenge is to choose a communication pattern that is neither too common (such as a latency or bandwidth benchmark) nor too esoteric (such as one so targeted to a single application that the results do not generalize to other patterns). Rather than create an appropriate problem ourselves, we borrow one from a set of exercises associated with a long-existing MPI tutorial [8]. This particular exercise, entitled "Exploring the cost of synchronization delays"[1] reads as follows (unedited):

> In this example, 2 processes are communicating with a third. Process 0 is sending a long message to process 1 and process 2 is sending a relatively short message to process 1 and then to process 0. Arrange the code so that process 1 has already posted an MPI_Irecv for the message from process 2 before receiving the message from process 0, but also ensure that process 1 receives the long message from process 0 before receiving the message from process 2.
>
> This seemingly complex communication pattern mimics a pattern that can occur in an application due to timing variations on each processor. If the message sent by process 2 to process 1 is short but long enough to require a rendezvous protocol, there can be a sigificant delay

[1] The problem statement and sample solution are available on the Web at `http://www-unix.mcs.anl.gov/mpi/tutorial/mpiexmpl/src3/3way/C/main.html`.

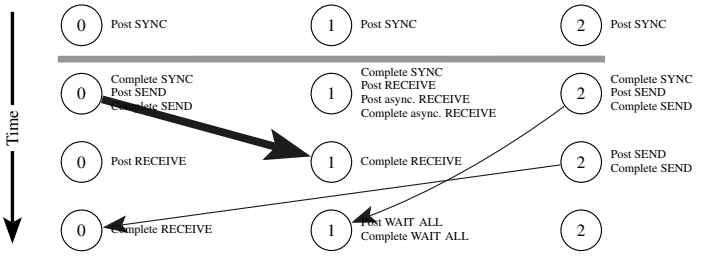


Fig. 1. Communication pattern described by the sample problem

before the short message from process 2 is received by process 1, even though the receive for that message is already available. Explore the possibilities by considering various lengths of messages.

In essence, this is the sort of performance test an application developer would create if his application uses such a communication pattern and he wants to find the source of its performance problems.

The sample solution, written in C and using MPI as the communication library, is 74 lines long and is illustrated in Fig. 1. The core communication and reporting routines are presented below but the reader is directed to the URL shown at the bottom of page 150 for the complete listing.

(44 lines omitted)

```
45      MPI_Barrier(MPI_COMM_WORLD);
46
47      if (myrank == 0) {
48          MPI_Send( bigdata, BIGSIZE, MPI_DOUBLE, 1, tag, MPI_COMM_WORLD);
49          MPI_Recv( litdata, litsize, MPI_DOUBLE, 2, tag, MPI_COMM_WORLD, &status);
50      }
51      else if (myrank == 1) {
52          MPI_Irecv(litdata, litsize, MPI_DOUBLE, 2, tag, MPI_COMM_WORLD, &request);
53          MPI_Send (litdata, litsize, MPI_DOUBLE, 2, tag, MPI_COMM_WORLD);
54          MPI_Recv( bigdata, BIGSIZE, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD, &status
                );
55          MPI_Wait( &request, &status );
56      }
57      else if (myrank == 2) {
58          MPI_Recv( litdata, litsize, MPI_DOUBLE, 1, tag, MPI_COMM_WORLD, &status);
59          t1 = MPI_Wtime();
60          MPI_Send( litdata, litsize, MPI_DOUBLE, 1, tag, MPI_COMM_WORLD);
61          ts1 = MPI_Wtime() - t1;
62          t1 = MPI_Wtime();
63          MPI_Send( litdata, litsize, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD);
64          ts2 = MPI_Wtime() - t1;
65      }
66
67      if (myrank == 2) {
68          printf("[%d] Litsize = %d, Time for first send = %f, for second = %f\n",
69                  myrank, litsize, ts1, ts2 );
70      }
```

(4 lines omitted)

There are a number of shortcomings of the preceding code:

1. Too little information is output. Without a more detailed record of the experimental setup (e.g., shared libraries and versions being used, compiler flags, environment variables, etc.) the application developer may overlook some performance-affecting detail. Ideally, much more information should be logged but doing so can require daunting coding effort.
2. 48 lines of initialization and finalization code (not shown) is too long. Application developers are unlikely to consider writing a special-purpose performance test if including header files, declaring variables, initializing the messaging layer, parsing the command line, and doing all of the other banal, non-performance-related activities take so much coding time.
3. The level of abstraction is too low. It is difficult to ascertain which lines of code correspond to which parts of the problem statement.

We attempt to address these shortcomings in Sects. 3 and 4.

3 A Library for Performance Testing

Regardless of what particular communication patterns or performance characteristics a benchmark tests, there are a number of mundane operations that the code must perform such as parsing the command line, recording elapsed time, computing performance statistics, and logging results to a file. Because such operations are of common utility, it is practical to implement them in a helper library so they can be reused by numerous performance tests.

For the pupose of this study, we use an existing library, the run-time performance library used by coNCePTuaL [9]. This library exports a rich set of functions intended to simplify performance testing. The reader is referred to the coNCePTuaL User's Guide [10] for details.

None of the performance library's functions are specific to any particular messaging layer. Consequently, the MPI calls in the code shown on page 151 remain intact while the library improves the following constructs:

Parsing the Command Line. Rather than explicitly scanning `argv[]` as in the original code, using `ncptl_parse_command_line()` provides error checking, support for short and long option names, and a `--help` option which describes each of the supported options.

Touching the Message Buffers. The original code writes dummy values to the message buffers as part of initialization. On CPUs with write-no-allocate cache policies, this will not yield the desired result of preloading the buffers into the cache. `ncptl_touch_data()`, in contrast, both reads and writes each word of the buffer. This is a prime example of the usefulness of a helper library: Once the library code is written, debugged, and made portable, all programs that link to the library automatically benefit.

Reading the Timer. The `ncptl_time()` function reads the highest-resolution timer available. Also, the log-file header includes the timer overhead, mean increment, and increment standard deviation, all of which are measured dynamically during initialization time [9]. Hence, unlike `MPI_Wtime()` – which, to begin with, is specific to MPI – the application developer knows exactly how reliable the platform's timing measurements can be.

Outputting Results. Replacing the original code's lone `printf()` with a set of `ncptl_log_`*something*`()` calls provides a number of benefits, the most important of which is that it helps make the performance test *reproducible*. A prior publication expands upon this issue and presents a sample log file [9] but the key idea is that by logging not only measurement data but also the entire experimental setup – system architecture, software used, timer accuracy, environment variables, etc. – log files become self-documenting. This is important to application developers because they can use a special-purpose performance test to diagnose a problem. Then, when they believe they have fixed the problem, they can re-run the performance test in exactly the same way that it was run previously and accurately compare the results.

The following shows how the original code's `printf()` can be replaced by library functions to improve program reproducibility:

```
                    (66 lines omitted)
67      if (myrank == 2) {
68          /* Log the results to a file. */
69          NCPTL_LOG_FILE_STATE *logfile = ncptl_log_open("syncdelays-conclib-%p.log",
                myrank);
70          ncptl_log_write_header (logfile, argv[0], "N/A", "N/A", myrank, numprocs, conc_args,
                1, NULL);
71          ncptl_log_write (logfile, 0, "Litsize", NCPTL_FUNC_NO_AGGREGATE, (double)
                litsize);
72          ncptl_log_write (logfile, 1, "Time for first send", NCPTL_FUNC_NO_AGGREGATE, (
                double)ts1);
73          ncptl_log_write (logfile, 2, "Time for second send", NCPTL_FUNC_NO_AGGREGATE
                , (double)ts2);
74          ncptl_log_commit_data (logfile);
75          ncptl_log_write_footer (logfile);
76          ncptl_log_close (logfile);
77      }
                    (4 lines omitted)
```

Using a performance-testing-centric library addresses shortcoming 1 on page 152. However, it does not address the remaining two shortcomings. This raises the question: Can special-purpose performance tests be developed more rapidly using a high-level language instead of C? The intention is to reduce code turnaround time and to relieve the application developer of the tedium of declaring variables, allocating and deallocating memory, and performing other low-level operations. To answer the preceding question, we re-coded the solution to the sample problem in Python – specifically, ScientificPython [11] for its MPI support – while continuing to use the performance library, which has a Python interface. The core communication and reporting routines are presented below:

(28 lines omitted)

```
29  comm_world.barrier()
30
31  if myrank == 0:
32      comm_world.send(bigdata, 1, tag)
33      comm_world.receive(litdata, 2, tag)
34  elif myrank == 1:
35      request = comm_world.nonblockingReceive(litdata, 2, tag)
36      comm_world.send(litdata, 2, tag)
37      comm_world.receive(bigdata, 0, tag)
38      request.wait()
39  elif myrank == 2:
40      comm_world.receive(litdata, 1, tag)
41      t1 = ncptl_time()
42      comm_world.send(litdata, 1, tag)
43      ts1 = ncptl_time() - t1
44      t1 = ncptl_time()
45      comm_world.send(litdata, 0, tag)
46      ts2 = ncptl_time() - t1
47
48  if myrank == 2:
49      # Log the results to a file.
50      logfile = ncptl_log_open("syncdelays-pyconclib-%p.log", myrank)
51      ncptl_log_write_header(logfile, sys.argv[0], "N/A", "N/A", myrank,
52                             numprocs, conc_args, 1, [])
53      ncptl_log_write(logfile, 0, "Litsize", NCPTL_FUNC_NO_AGGREGATE, litsize)
54      ncptl_log_write(logfile, 1, "Time for first send", NCPTL_FUNC_NO_AGGREGATE, ts1)
55      ncptl_log_write(logfile, 2, "Time for second send", NCPTL_FUNC_NO_AGGREGATE, ts2
            )
56      ncptl_log_commit_data(logfile)
57      ncptl_log_write_footer(logfile)
58      ncptl_log_close(logfile)
```

Line for line, the Python version is fairly similar to the C version; an application developer unfamiliar with Python should have no trouble understanding the code. However, we can say that the Python version remedies shortcoming 2 on page 152 by being shorter (by ∼21%) and arguably less error-prone than the C version. Nevertheless, the level of abstraction is unchanged from the C version; the connection between the problem statement and the code remains unclear. In Sect. 4 we determine if a domain-specific language can improve the situation.

4 A Domain-Specific Language for Performance Testing

Application developers – and programmers in general – are often loath to use domain-specific languages. There is an inherent cost to learning a new language which can be hard to justify in absense of *a priori* understanding of the domain-specific language's practical benefits. It is therefore important for this paper to evaluate how well a domain-specific language can be suited to the rapid development of application-specific network performance tests.

coNCePTuaL – whose unusual capitalization stands for "Network Correctness and Performance Testing Language" – is a domain-specific language designed to facilitate the rapid development of special-purpose network performance tests [9]. The coNCePTuaL language is English-like and uses SPMD semantics to express parallelism. coNCePTuaL programs implicitly use the

performance library described in Sect. 3 through various language idioms. The following is the complete, commented coNCePTuaL solution to the problem specified in Sect. 2:

```
1  Require language version "0.5.2b".
2
3  bigsize is "Size of big message (doubles)" and comes from "--bigsize" or "-b" with
       default 10000.
4  litsize is "Size of small message (doubles)" and comes from "--litsize" or "-n" with
       default 1.
5
6  Assert that "this program must be run with at least 3 processes" with num_tasks>=3.
7
8  All tasks touch all message buffers then
9  all tasks synchronize then
10
11 # "Arrange the code so that process 1 has already posted an MPI_Irecv
12 # for the message from process 2 before receiving the message from
13 # process 0,"
14 task 1 asynchronously receives a litsize doubleword message from task 2 then
15
16 # "Process 0 is sending a long message to process 1"
17 task 0 sends a bigsize doubleword message to task 1 then
18
19 # "also ensure that process 1 receives the long message from process 0
20 # before receiving the message from process 2."
21 task 1 awaits completion then
22
23 # "process 2 is sending a relatively short message to process 1 and
24 # then to process 0."
25 task 2 resets its counters then
26 task 2 sends a litsize doubleword message to unsuspecting task 1 then
27 task 2 logs litsize as "Litsize" and elapsed_usecs as "Time for first send" then
28 task 2 resets its counters then
29 task 2 sends a litsize doubleword message to task 0 then
30 task 2 logs elapsed_usecs as "Time for second send".
```

Two things are immediately apparent about the preceding code listing. First, its length is half that of even the corresponding Python code. Second, coNCePTuaL statements closely correspond to the natural-language statements in the problem specification. For example, posting a send implicitly posts the matching receive (suppressable with the **unsuspecting** keyword, as in line 26), as this is how network performance tests typically are described textually. By raising the abstraction level of performance tests to match that of a natural-language problem specification, coNCePTuaL satisfactorily addresses shortcoming 3 on page 152, the final shortcoming of the original C solution. There is no performance penalty to using coNCePTuaL, as indicated by the following message overheads output by the various code versions: C: (5µs, 2µs); C + library: (4µs, 1µs); Python: (10µs, 5µs); coNCePTuaL: (4µs, 2µs). (coNCePTuaL reports slightly lower overheads than C because it directly reads the CPU cycle counter; when configured to use `gettimeofday()` it reports the same overheads as C.)

coNCePTuaL is not limited to MPI. Any of a variety of code generators is selectable at compile time. In fact, Fig. 1 was produced automatically from the preceding listing merely by specifying an "illustration" code generator.

5 Conclusions

Special-purpose network performance tests enable an application developer to experiment with communication alternatives faster and with less hassle that would be required to restructure the original application. Such tests are rarely used in practice, however, because of the their development overhead. This paper followed the evolution of a sample special-purpose performance test from its natural-language problem specification through C, C + helper library, Python + helper library, and domain-specific-language solutions. Each alternative improved upon the previous one: richer output; shorter, easier-to-develop programs; and, higher levels of abstraction. However, only the domain-specific language, coNCePTuaL, fully encapsulated all three of those benefits.

This paper elucidated the tradeoffs that application developers face when considering using special-purpose network performance tests as a tool for performance optimization. Developers can reduce development time with little effort merely by calling appropriate library functions. Developers who couple a library with a high-level language can further reduce development time, making special-purpose network performance tests more attractive to write. The biggest payoff comes from using a domain-specific language, which provides a natural mapping from problem specification to working code while still retaining the benefits of short code lengths, ease of development, and reproducibility of results.

The coNCePTuaL compiler and performance library are freely available from `http://conceptual.sourceforge.net/`.

References

1. Bleck, R.: An oceanic general circulation model framed in hybrid isopycnic-Cartesian coordinates. Ocean Modelling **4** (2002) 55–88
2. Blelloch, G., Narlikar, G.: A practical comparison of N-body algorithms. In: Parallel Algorithms. Volume 30 of DIMACS: Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, DIMACS (1997)
3. Basney, J., Raman, R., Livny, M.: High throughput Monte Carlo. In: Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing, San Antonio, Texas (1999)
4. Snell, Q.O., Mikler, A.R., Gustafson, J.L.: NetPIPE: A network protocol independent performance evaluator. In: Proceedings of the 1996 ISMM International Conference on Intelligent Information Management Systems, Washington, DC, ACTA Press (1996)
5. Gropp, W., Lusk, E.: Reproducible measurements of MPI performance characteristics. In: Proceedings of the 6th European PVM/MPI Users' Group Meeting (EuroPVM/MPI'99). Volume 1697 of Lecture Notes in Computer Science., Barcelona, Spain, Springer-Verlag (1999) 11–18
6. Pallas, GmbH: Pallas MPI Benchmarks—PMB, Part MPI-1. (2000)

7. Reussner, R., Sanders, P., Prechelt, L., Müller, M.: SKaMPI: A detailed, accurate MPI benchmark. In: Proceedings of the 5th European PVM/MPI Users' Group Meeting (EuroPVM/MPI'98). Volume 1497 of Lecture Notes in Computer Science., Liverpool, United Kingdom, Springer-Verlag (1998) 52–59
8. Gropp, W.: Tutorial on MPI: The Message-Passing Interface. Argonne National Laboratory, Argonne, Illinois. (1995) Available from `ftp://info.mcs.anl.gov/pub/mpi/tutorial.ps`.
9. Pakin, S.: Reproducible network benchmarks with coNCePTuaL. In: Proceedings of the 10th International Euro-Par Conference. Volume 3149 of Lecture Notes in Computer Science., Pisa, Italy, Springer (2004) 64–71 ISBN 3-540-22924-8. Available from `http://www.c3.lanl.gov/~pakin/papers/europar2004.pdf`.
10. Pakin, S.: coNCePTuaL user's guide. Technical Report LA-UR 03-7356, Los Alamos National Laboratory, Los Alamos, New Mexico (2004) Available from `http://www.c3.lanl.gov/~pakin/software/conceptual/conceptual.pdf`.
11. Hinsen, K.: ScientificPython User's Guide. Centre National de la Recherche Scientifique d'Orléans, Orléans, France. (2002)

Providing Interoperability for Java-Oriented Monitoring Tools with JINEXT

Włodzimierz Funika and Arkadiusz Janik

Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
funika@uci.agh.edu.pl

Abstract. While Java is getting an increasingly widely used programming language, Java Virtual Machine has become an important platform for networking and distributed computing. Due to the increasing complexity of programs, the demand for monitoring tool support (debuggers, performance analyzers etc.) for efficient computing is growing as well. A special, open interface J-OMIS, which provides an efficient support for monitoring distributed Java programs, is derived from the On-line Monitoring Interface Specification (OMIS) that enables to use multiple monitoring tools simultaneously. OMIS has been developed with interoperability in mind but not all structural and logical conflicts have been solved. Addressing the missing aspects of interoperability support within J-OMIS is intended to increase the simplicity of developing the monitoring tools which will synergetically support each other. In the paper we present the concept of JINEXT, an extension to OMIS, which is aimed to provide interoperability between monitoring tools.

1 Introduction

In the world of nowadays' computer science creating new applications has become a very complicated, difficult, often long-term process. The software systems become larger because the complexity of problems solved with them grows up. In order to simplify, accelerate as well as to minimize the number of errors and bugs done during developing process, it is necessary to use *monitoring tools* supporting this process. Interoperability is the ability of two or more software systems to cooperate with each other. However, using tools from different vendors may not accelerate the developing process but even disrupt it, unless a special coordinating facility, like *monitoring system* provides a kind of interoperability support.

In this paper we present some issues of adding an interoperability-oriented extension to Online Monitoring Interface Specification (OMIS), which solves structural and logical interoperability conflicts not handled by OMIS. In Section 2, we focus on the *mediator* interoperability model. In Section 3, we address the common problems connected with the issue of interoperability . Section 4 presents JINEXT, an extension to the OMIS specification and in Section 5 we present a practical example of using JINEXT. In Section 6 we discuss the ongoing work on JINEXT and further research.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 158–165, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Related Work

There are a number of different concepts of *interoperability* [10]. In the paper we address interoperability in the context of monitoring tools, as a kind of *cooperation* between them on the *semantical* level where interpretation of actions and events provided by each cooperating tool must be guaranteed.

A number of different interoperability models can be found in literature. In the *ToolTalk* system [1] a tools' interaction can be described independently from the underlying communication mechanism. However, debugging tools which use ToolTalk, as well as difficult deadlock analysis are big disadvantages of it. The Eureka Software Factory (ESF) and Purtillo [1] based on the *hardware metaphore* model use a special language to describe mappings between logical modules (tools) of the system. These systems solve interoperability problems on the *control* level, so the semantical issues are out of their interests. There is also a model based on the solution where tools are interpreted as the agent systems. An example is LARKS [6]. There is a specialized layer (called *matchmarker*), used as an intermediator between the tools. The Agents Capability Description Language (ACDL) [6] is a language used to describe services provided by different tools. A description involves both, semantical and structural aspects of a service, but conflicts due to the issue of interoperability are still not solved.

The *mediator* model stresses the separation between a behavior and tools which implement this behavior [3]. A *behavioral relationship* term is used to describe a relation between different tools, which is a connection between two behaviors of two different tools. Each behavior in the mediator model is realized by a proper abstract behavioral type (ABT) [2]. For instance, an editor behavior is realized by *Editor* ABT with an operation to save a file and an event *Saved()*. This event is raised whenever a file is saved. A behavioral relationship is realized by the mediator. Having registered the *UponSave()* operation with this event, it is invoked whenever the event is raised. Note that the mediator is external to the objects whose behaviors are integrated. The editor is viewed as an object which provides the *Save()* action and generates *Saved()* event. A notification about saving a file is done by the broadcast message. The Mediator "captures" this message and calls the compiler's *Compile()* method.

The mediator solution promotes software modularization and is resistant to a behavior evolution. For example, if we would like to improve the behavior of compiler so it recompiles a modified file only when CPU load is low, only the behavioral relationship can be changed. The editor and compiler do not have to be changed. The only modification is an update of the mediator object. Moreover, if we want to send the modified file to a remote backup directory whenever it is modified, we can introduce an additional tool, e.g. a *backup manager*. A way to improve the behavioral model is to add another relation between the improved backup manager and the editor. To implement this change, we have to add a backup manager object and the mediator realizing the relation between the manager and the editor [3].

3 Common Aspects of the Issue of Interoperability

When considering interoperability as the ability of two or more applications to cooperate, we will concentrate on a special case of software systems, called *monitoring tools*, as run-time applications which are used to observe and/or manipulate the execution of a software system. Monitoring tools need a *monitoring system* to be able to work in a proper way based on monitoring data. The monitoring system is part of the tool infrastructure responsible for observing and manipulating the target system, which provides the monitoring functionality for tools, based on some interface. OMIS specifies an interface between a monitoring system and tools [4].

In this context, interoperability is the ability of applying two or more tools to the same software system, at the same time. In order to provide these features some common problems have to be solved.

Structural Conflicts. The first problem is to run the tools *concurrently*, which means to allow them to work at the same time being attached to the same software system. The monitoring system should provide a mechanism for the concurrent operation of monitoring tools, for sharing the common resources (hardware, processes, threads, classes, objects, memory). Without a common monitoring system the tools may access a target system in a completely uncoordinated manner [2].

Logical Conflicts. *Logical conflicts* are connected with semantical issues. As long as logical conflicts are not solved tools cannot co-exist *consistently*. It means that when manipulating on the target system, tools cannot preserve a consistent view of this system, e.g. when a profiler measures the time of a process execution while this process is suspended by a debugger. More formally, the consistency problem can be described as a *read/write access conflict* [2].

4 OMIS Versus the Interoperability of Tools

OMIS and the OCM monitoring system, its reference implementation, are intended to provide a kind of interoperability for tools relating to structural conflicts and conflicts on exclusive objects [5]. All tools that share an object use the same controlling monitor process that coordinates accesses to the monitored objects. Therefore, tools are enabled to coexist. However, the transparency problem was not resolved yet. The support for avoiding logical conflicts in OMIS is incomplete.

The concept of OMIS allowed to extend OMIS and the OCM by monitoring support for Java distributed applications, known as J-OMIS and J-OCM, respectively [7]. To provide a wide interoperability for Java-oriented tools, we designed Java INteroperability EXTension (JINEXT) as an extension to OMIS (see Fig. 1).

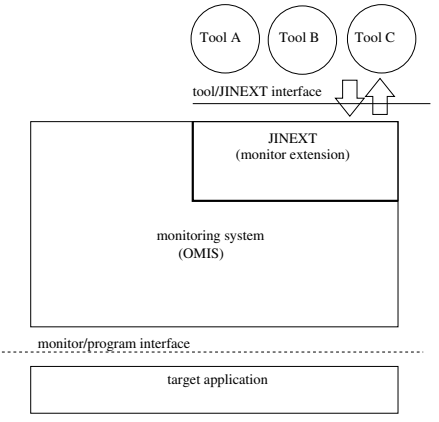


Fig. 1. The overview on JINEXT architecture

In order to be interoperable some conditions have to be met:

- the tools communicate with the JINEXT extension only, instead of calling the monitoring system directly;
- a component implementing the JINEXT specification (a.k.a. *JINEXT compliant system*) is responsible for passing requests to the monitoring system and for synchronization between different tools, as well as for choosing target monitoring tools to be informed about changes due to request;
- each tool must implement a single *interface* (*events* and *actions*) which is used to emphasize that the behavior and functionality provided by each tool is the same for all the tools of a given type (e.g. there can be four different debuggers, developed by different providers, in different languages, all of them sharing a common interface).

The Mediators. *The mediators* in JINEXT follow the *mediator* interoperability model [3]. A mediator is a connection between the event generated by a tool and the action which should be executed by another tool whenever the event occurs. In JINEXT each mediator is used to describe one *behavioral relation* between an event and an action. If the developer wants to introduce a tool implementing a new tool type, she/he has to provide mediators which are responsible for interoperability between the new tool type and all other tool types.

The Mediator Function. While the mediator in JINEXT refers to an abstract behavioral relation only, the *mediator function* implements this relation. A mediator defines that if tool A raises event X then tool B should execute action Y while a mediator function is responsible for translating the parameters of event X into parameters of action Y.

The important advantage of the model used in JINEXT is the ease of adding a new relation between the monitoring tools while their implementation stays

unmodified. Moreover, the mediator function makes it possible to change the behavior of tool B (used in above example) while the mediator remains the same. It means that monitoring tool's behavior can be flexibly changed by providing new mediator functions, while the very mediators remain untouched, after they have been once well defined.

There are two different modes of tools' cooperative in JINEXT: *non-cooperative mode* and *cooperative mode*.

- *non-cooperative mode* - it is defined w.r.t. a pair of monitoring tools what means that these two tools can work in the non-cooperative mode when considering this particular pair but any of these tools can work in the cooperative mode with some other tool. In the *non-cooperative mode* tools working with the JINEXT compliant system do not have to know that there are other monitoring tools connected to the system, which are monitoring the target application. JINEXT takes care of tools' synchronization.
- *cooperative mode* - two tools work in this mode if they know about each other. Each of them can request another tool for an action execution. It means that the functionality provided by the first tool can be used by the second one, e.g. to extend the functionality of monitoring tools and to build a complex toolset consisting of smaller components, like the *IDE* (Integrated Developer's Environment).

5 Use Case

In order to verify the features of JINEXT we have developed its prototype implementation, called JINTOP. We have also developed four example monitoring tools: an editor, a compiler, a debugger and a profiler. A sample scenario of interoperability between tools is as follows:

1. the user changes the source file of the application *App*
2. the **editor** *saves* the changes;
3. the **compiler** *recompiles* the code and generates the new binary file;
4. the simultaneously used **debugger** *restarts* the debugged application with a new code;
5. during the restart of the application **profiler** *pauses* and *resumes* the previously defined performance measurements.

The sequence diagram for this scenario explaining the role of JINTOP is presented in Fig. 2.

The editor notifies JINTOP about event *source_code_saved* and JINTOP uses mediators to obtain information on the attached tools which are interested in this event. Each event is checked in the context of calling tool so two events e.g. *Debugger.process_stopped* and *Profiler.process_stopped* will be handled by different mediators and may trigger different actions. After having checked events, *callback* method *jinext_action()* is used to execute a tool's action (e.g. *recompile_source_code*). This causes raising the *binary_code_modified* event. If a tool

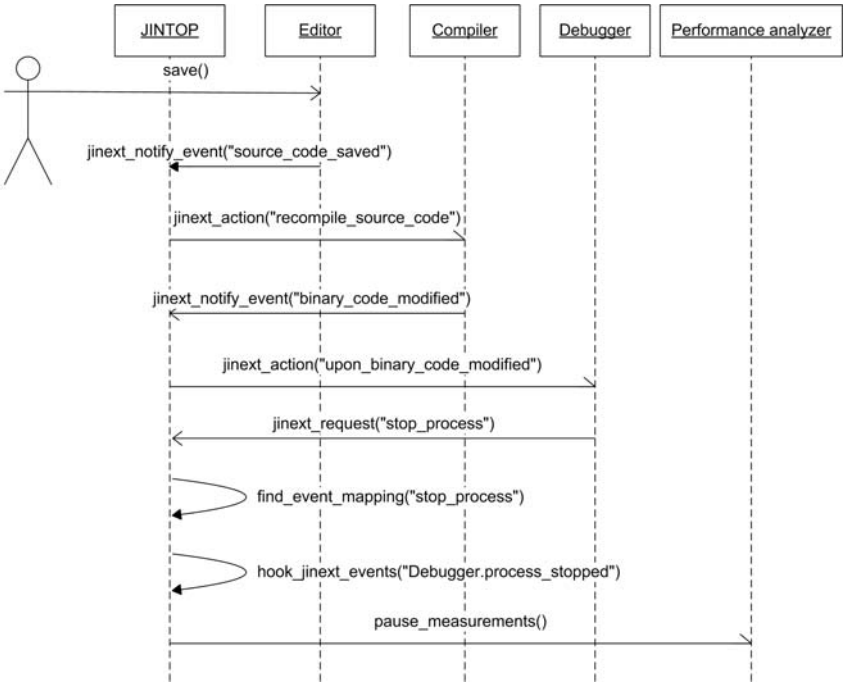


Fig. 2. The sequence diagram for the *save()* operation

requests OMIS/J-OMIS service (e.g. a debugger requesting the *stop_process* service) JINTOP executes this request and hooks zero or more events (*Debugger.process_stopped*) which trigger (via mediators) zero or more actions (*Performancer.pause_measurements*). As one can see, events can be hooked directly by tools (*jinext_notify_event*) or by JINTOP as a result of executing an OMIS/J-OMIS service. JINTOP takes care of consistency after having executed the OMIS/J-OMIS service by the tool. Moreover, JINTOP do not bother any tools which are not attached to the *tokens* (threads, processes, classes, objects etc.) connected with the notified event.

Overhead Due to Using JINTOP. We have carried out experiments to measure the overhead induced due to using JINTOP instead of direct calls to the OCM. The results of our experiment are presented in Fig. 3. The experiment was done on AMD Athlon 2000+ 1,8GHz with 1GB of RAM, under Linux 2.23 OS.

The first observation is that the execution time of processing a request is longer if there are more processes the request refers to. The second conclusion is that the overhead introduced by JINTOP increases if there are more monitoring tools registered by JINTOP. However, it is always less than ca 22% and can be further reduced by changing the internal data structures of JINEXT implementation.

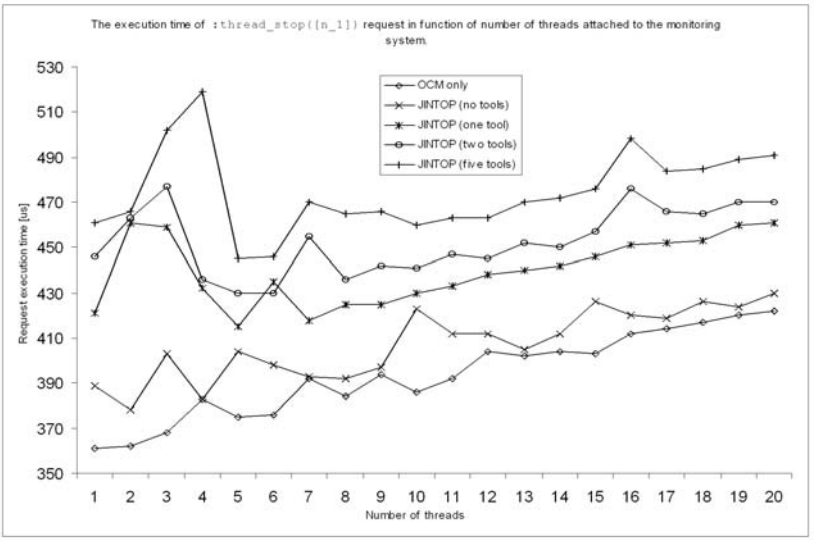


Fig. 3. The execution time of an example request in function of the number of threads attached to the monitoring system

6 Conclusion and Future Work

JINEXT, as an extension to OMIS for tools' interoperability, is designed to enable different monitoring tools to run concurrently, via solving logical and structural conflicts. It enables simultaneous operation of different monitoring tool types, without system failures. The mediator interoperability model used in JINEXT guarantees that an evolution of a behavior of monitoring tools can be done seamlessly from the viewpoint of interoperability.

The first version of JINEXT has been released. However, some useful services should also be added to JINEXT, e.g. these ones which enable dynamic addition of a new tool type allowing the developer to describe the monitoring tool type in an external XML file. The next step in further research is to extend JINEXT to be compatible with OCM-G [8]. Providing the interoperability of monitoring tools working in a grid environment is necessary if we want to allow more than a single user to work on the same target object from within different nodes or sites. Let us consider the situation when one user on node A is doing some performance measurement on an application whereas another user on node B is debugging this application. If these monitoring tools are not interoperable the first user will receive false measurement results. A solution would be to add a special interoperability agent on each node of the grid. The agent analyzes requests and recognizes whether they are coming from a local or remote node of the grid. In the second case, the agent will do all necessary token translations. The agent can also support an additional interoperability-related security policy.

Acknowledgements. Our thanks go to Prof. Roland Wismüller for valuable discussions. This research was partially supported by the KBN grant 4 T11C 032 23.

References

1. Bergstra, J. A., Klint P.: The ToolBus - a component interconnection architecture, Programming Research Group, University of Amsterdam, Meeting, Band 1697 aus Lecture Notes in Computer Science, page 51–58, Barcelona, Spain, September 1999. ftp://info.mcs.anl.gov/pub/techreports/reports/P754.ps.Z
2. Roland Wismüller: Interoperable Laufzeit-Werkzeuge für parallele und verteilte Systeme, Habilitationsschrift, Institut für Informatik, Technische Universität München, 2001
3. K. J. Sullivan. Mediators: Easing the Design and Evolution of Integrated Systems. PhD. thesis, Dept. of Computer Sciences and Engineering, Univ. of Washington, USA, 1994. Technical Report 94-08-01. ftp://ftp.cs.washington.edu/tr/1994/08/UW-CSE-94-08-01.PS.Z
4. Ludwig, T., Wismüller, R., Sunderam, V., and Bode, A.: OMIS – On-line Monitoring Interface Specification (Version 2.0). Shaker Verlag, Aachen, vol. 9, LRR-TUM Research Report Series, (1997) http://wwwbode.in.tum.de/~omis/OMIS/Version-2.0/version-2.0.ps.gz
5. Roland Wismüller: Interoperability Support in the Distributed Monitoring System OCM, In: R. Wyrzykowski et al., (eds.), Proc. 3rd International Conference on Parallel Processing and Applied Mathematics - PPAM'99, pages 77-91, Kazimierz Dolny, Poland, September 1999, Technical University of Czestochowa, Poland. Invited Paper.
6. Katia Sycara, Jianguo Lu, Matthias Klusch: Interoperability among Heterogenous Software Agents on the Internet, The Robotics Institute Carnegie Mellon University, Pittsburgh, USA, October 1998
7. M. Bubak, W. Funika, M.Smętek, Z. Kiliański, and R. Wismüller: Architecture of Monitoring System for Distributed Java Applications. In: Dongarra, J., Laforenza, D., Orlando, S. (Eds.), Proceedings of 10th European PVM/MPI Users' Group Meeting, Venice, Italy, September 29 - October 2, 2003, LNCS 2840, Springer, 2003
8. Baliś, B., Bubak, M., Funika, W., Szepieniec, T., and Wismüller, R.: An Infrastructure for Grid Application Monitoring. In: Kranzlmüller, D. and Kacsuk, P. and Dongarra, J. and Volkert, J. (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, 9th European PVM/MPI Users' Group Meeting, September - October 2002, Linz, Austria, 2474, Lecture Notes in Computer Science, 41-49, Springer-Verlag, 2002
9. Günther Rackl: Monitoring Globus Components with MIMO, Institut für Informatik, PhD Thesis, Technische Universität München, March 2000
10. Peter Wegner: Tutorial Notes: Models and Paradigms of Interaction. Technical Report CS-95-21, Department of Computer Science, Brown University, Providence Rhode Island 02912, USA, September 1995 http://www.cs.brown.edu/publications/techreports/reports/CS-95-21.html

RDVIS: A Tool That Visualizes the Causes of Low Locality and Hints Program Optimizations

Kristof Beyls, Erik H. D'Hollander, and Frederik Vandeputte

Departement of Electronics and Information Systems,
Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
{kristof.beyls, erik.dhollander, frederik.vandeputte}@elis.ugent.be

Abstract. The visualization tool `rdvis` is presented which aims at helping the programmer to find program transformations to improve temporal data locality. We present a number of locality metrics that capture the necessary information. Based on a cluster analysis of basic block vectors, the tool gives strong hints on which program transformations are needed. The visualizer allowed us to find the necessary transformations for three SPEC2000 programs in just a few minutes. After performing these transformations, the programs run 3 times faster on average on a number of different platforms.

1 Introduction

For many applications, half of their execution time the processor performs no useful work due to data cache misses. Since the majority of the cache misses result from poor temporal or spatial locality, this bottleneck can be removed by increasing locality. In this paper, we propose a visualization of the causes of low locality. In contrast to earlier visualizations of cache misses[1, 4, 8, 10, 11, 13, 15, 16, 17], the proposed visualization is expected to give more useful hints on how to transform the code to improve locality and eliminate cache misses in a platform-independent way.

Whenever a memory location is used multiple times throughout the program execution (i.e. it is *reused*), the corresponding data may stay in the cache between the different accesses to it, and cache hits may result. However, when the reuses are separated by accesses to a lot of different data, the probability that the data remains in the cache between use and reuse is low. The number of different memory locations accessed between use and reuse is called the *reuse distance*[3].

The proposed visualization indicates the constructs in the source code that induce these low-locality reuses. By also highlighting the code that is executed between use and reuse, the programmer sees which code is responsible for increasing the reuse distance of a given pair of references. As such, the visualization shows the code that should be restructured to decrease reuse distances.

In a previous paper [2], we showed that this information helps the programmer to optimize programs platform-independently with an average speedup of 3 and a maximum of 11. However, the visualization was not as extensive as described in this paper, and more than a day of work was needed to understand

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 166–173, 2005.
© Springer-Verlag Berlin Heidelberg 2005

each application's bottlenecks. With the visualization presented here, the application's bottlenecks can be understood in just a few minutes. In comparison to earlier work, we present the following novelties:

- The GCC compiler has been adapted to instrument programs to measure reuse distance and the code that is executed between reuses and to record the exact locations in the source code that lead to memory accesses.
- An automatic visualizer has been developed.
- The visualization might show a lot of different reference pairs that lead to long distance reuses. However, we found that many different long reference pairs can often be optimized by a single program transformation. Previously, it was unclear which reference pairs could be optimized by the same program transformation. In this paper, we propose a clustering algorithm, based on basic block vectors (BBV). In contrast to other work using BBVs [14] where BBVs indicate the basic blocks executed in a fixed-length interval of a program execution, our BBVs represent the code executed between data reuses.

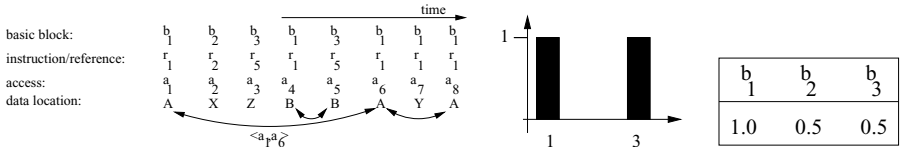
In Section 2, the proposed method for improving locality is presented. Section 3 presents the visualizations to support the programmer in effectively following the method. Examples of the visualization are discussed in Section 4, and concluding remarks follow in Section 5.

2 Method for Improving Data Locality

In this section, the visualization-based method presented in [2] to improve data locality is briefly presented. It focuses on resolving capacity misses, the dominant kind of misses for most applications and cache configurations [5].

Quantifying Poor Locality. A capacity miss occurs when too many different memory locations are accessed between reuses of the same data. This is described more formally in the definition and lemma below.

Definition 1. *A* **memory reference** *corresponds to a read or a write in the source code, while a particular execution of that read or write at runtime is a* **memory access***[7]. A* **reuse pair** $\langle a_1, a_2 \rangle$ *is a pair of memory accesses in a memory access stream, which touch the same memory location, without intermediate accesses to that location. The* **reuse distance** *of a reuse pair* $\langle a_1, a_2 \rangle$ *is the number of unique memory locations accessed between accesses* a_1 *and* a_2*. A* **reference pair** $\langle r_1, r_2 \rangle$ *is a pair of memory references. The histogram of all reuse distances of the reuse pairs* $\langle a_1, a_2 \rangle$*, for which* a_1 *is an access produced by* r_1*, and* a_2 *is produced by* r_2*, is called the* **reference pair's reuse distance histogram RDH(**$\langle r_1, r_2 \rangle$**)**. *The* **basic block vector of a reference pair** $\langle r_1, r_2 \rangle$ **BBV(**$\langle r_1, r_2 \rangle$**)** *is a vector which contains real numbers between 0 and 1, each corresponding to a basic block in the program. The value of an element is the fraction of reuse pairs in* $\langle r_1, r_2 \rangle$ *for which the basic block is executed between use and reuse. (see Fig. 1)* □



(a) access stream (b) RDH($\langle r_1, r_1 \rangle$) (c) BBV($\langle r_1, r_1 \rangle$)

Fig. 1. Example memory access stream. In (a), the reuse pairs are indicated by arrows. The second row indicates 8 memory instructions, generating the memory accesses in the third row. The top row indicates the basic blocks containing the references in the second row. The bottom row shows the corresponding addresses. The reuse distance of $\langle a_1, a_6 \rangle = 3$. (b) shows RDH($\langle r_1, r_1 \rangle$), while (c) shows BBV($\langle r_1, r_1 \rangle$)

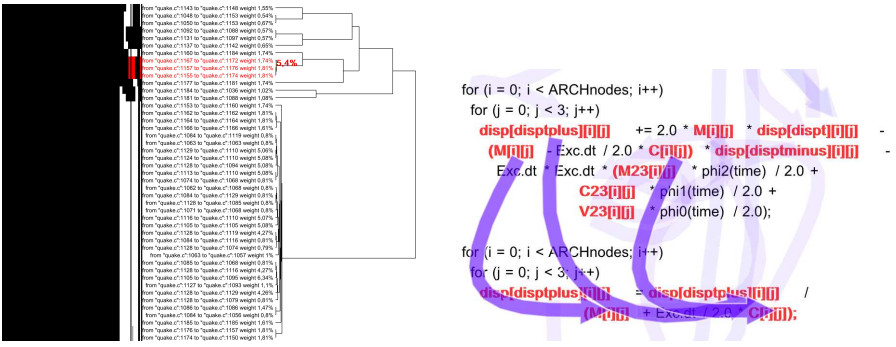
Lemma 1. *In a fully associative LRU cache with n lines, a reuse pair* $\langle a_1, a_2 \rangle$ *with reuse distance* $d < n$ *will generate a hit at access* a_2*. If* $d \geq n$*,* a_2 *misses the cache.* □

Program Transformations to Improve Data Locality. The locality measurement and visualization is performed automatically. Based on the visualization, the capacity misses can be minimized by the programmer. The most important way to eliminate capacity misses is to reduce the distance between use and reuse for long reuse distances, so that it becomes smaller than the cache size. This can be done by reordering computations (and memory accesses), so that *higher temporal locality* is achieved. Examples of such optimizations are loop fusion and loop tiling[12]. The general idea behind this family of optimizations is to try to do as much useful computations as possible on the data while it is in the cache.

3 Support Provided by the Rdvis Tool

3.1 Instrumentation

The reference pairs, their reuse distance histogram and the code executed between use and reuse is measured by instrumenting the program. The instrumentation is implemented in the GCC compiler. The C front end has been adapted to record the precise location of memory references in the source code. The instrumentation occurs in an early stage of the compilation process so that the source code information is still available. All the pointer dereferences and array accesses are assumed to lead to load and store instructions in the resulting binary. It is assumed that all scalar variables can be allocated to registers, so they don't lead to memory accesses. Even when the register file of the processor is too small to contain all scalar variables, the accesses are very likely to have very good locality, so it is not necessary to take them into account. At the end of program execution all the recorded reuse pairs, the associated distance histograms and BBVs of intermediately executed code are written to a file.



(a) Cluster analysis view of Equake. A cluster of three reference pairs has been selected, which together produce 5.4% of all capacity misses. As a result, the corresponding arrows are highlighted in Fig. 2(b). The same clustering, with a different selected cluster is shown zoomed-in in Fig. 4

(b) Since the only highlighted code is in the two consecutive loops, it can be concluded that the use is in the first loop nest and the reuse is in the consecutive execution of the second loop nest. Therefore, the use and reuse can be brought closer together by loop fusion

Fig. 2. Cluster of reference pairs that can be optimized using loop fusion

3.2 Visualization

The visualizer is written in Java and reads the recorded reuse information. The following paragraphs describe the two different kinds of visualization.

Reference Pair Visualization. The reference pairs are visualized by drawing an arrow from the first reference to the second reference in the source code. Each arrow has a weight, which is the percentage of capacity misses induced by the corresponding reference pair. Only the heaviest arrows are drawn, so that the drawn arrows together represent at least 90% of all capacity misses. An example of such arrows are found in Figures 2(b) and 3. The color of the arrows corresponds to their weight, ranging from red for heavy-weighted arrows to blue for light-weighted arrows. The transparency of the arrows allows to better see the code covered by the arrows. Only the arrows that are of interest (see cluster analysis below) are clearly visible. The other arrows are made highly transparent, to make them nearly invisible, so that the users attention is not disturbed by the arrows that are not currently under investigation.

When the arrow is clicked by the mouse a pop-up menu lets the user choose between showing the reuse distance histogram of that reference pair, or highlighting the memory references that are executed between the use and the reuse. An example of such highlighted memory references can be found in Figures 2(b) and 3.

Clustering of Reference Pairs with Large Reuse Distances. After seeing the reuses at long distances, the programmer needs to think about how to reduce the reuse distance. He can do this by using program transformations that reduce the amount of data accessed between use and reuse. For the programs that we visualized, we found that typically one or two program transformations improve the majority of long-distance reuses, even though up to 30 different reference pairs are found in the 90th percentile of long distance-reuses. So, different reference pairs can be optimized by the same program transformation. Since the programmer should focus on the needed program transformations, and not on individual reference pairs, the visualizer clusters the pairs, so that those in the same cluster are likely to be resolved by the same program transformation.

The basic assumption in the clustering of reference pairs is that to diminish the long distance reuses, memory accesses executed between use and reuse should be moved either before the use or after the reuse. When different reference pairs have the same code executed between use and reuse, they are very likely to need the same program transformation to have less data accessed between the reuses. Therefore, for each reference pair, the basic block vector is recorded, which represents the code that is executed between use and reuse. Then, these BBVs are clustered, using a hierarchical agglomerative clustering algorithm[9], where the distance between two basic block vectors is determined by the Manhattan metric[14]. Examples of the visual representation of the computed hierarchical clusters are shown in Figures 2(a) and 4. In the middle, a textual description of each reference pair is shown. On the right a dendrogram is shown which graphically indicates the distance between subclusters. On the left, the basic block vectors for each reference pair is shown. A black pixel means that the corresponding basic block is never executed between use and reuse, while a white pixel indicates that the corresponding block is always executed between use and reuse. Based on this visualization, the user selects interesting subclusters, by clicking a subcluster in the dendrogram shown on the right hand side.

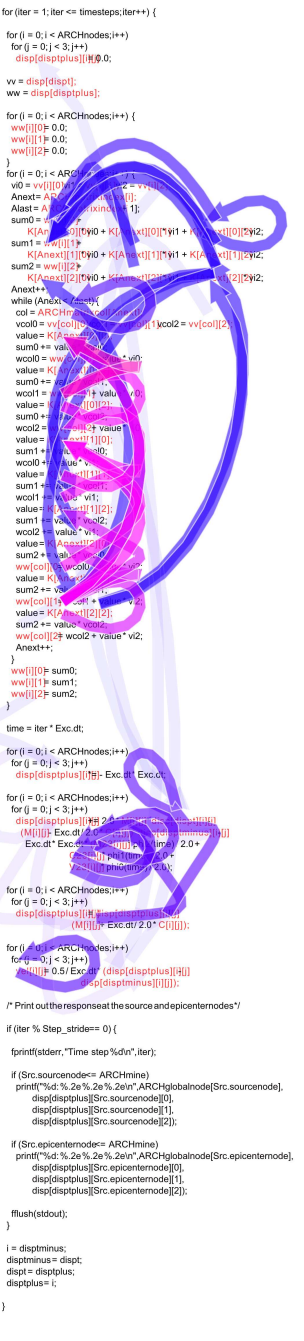


Fig. 3. Highlighted arrows corresponding to the selected cluster in Fig. 4

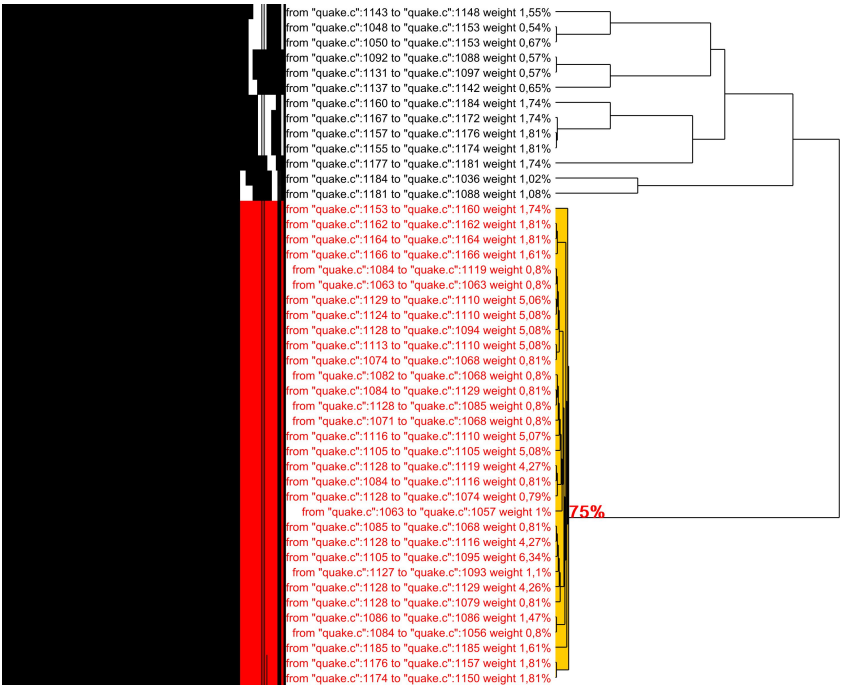


Fig. 4. The main cluster of Equake. The corresponding arrows and intermediately executed code is shown in figure 3

After selecting a subcluster the corresponding BBVs are highlighted, the percentage of long distance reuses generated by that cluster is shown in the dendrogram, and in the source code window, the corresponding arrows are highlighted, together with the memory references that are executed between uses and reuses. The other arrows are made highly transparent, so that they don't distract the users attention away from the currently investigated cluster.

4 Experiments

In [2], the three SPEC2000 programs with the largest cache bottleneck (Mcf, Equake, Art) were optimized, based on a manual drawing of the arrows corresponding to the most important reference pairs [2]. After manually looking for code that is executed between use and reuse, which took more than a day of work, program transformations were found to reduce the number of long reuse distances in Art and Equake. Mcf proved too difficult to analyze by hand, and only prefetch instructions could be inserted to try to hide the cache miss latency. As can be seen in Fig. 5, the optimized versions of Art and Equake, where the number of long reuse distances was reduced, gives consistent speed-up on multiple platforms. However, inserting prefetch instructions in Mcf does not reduce cache misses,

and the optimization is not profitable on all platforms. In the rest of this section, we show how the tool can help identify the needed program transformations in just a couple of minutes.

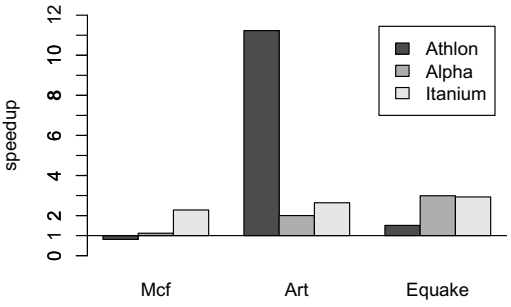


Fig. 5. Speedups after program transformations based on visualization of reference pairs

Loop Fusion. Based on the cluster analysis of the Equake program, as seen in Fig. 2(a), one can select the highlighted cluster in the figure, to see how this cluster of reference pairs could be optimized. When the cluster is selected, the corresponding arrows are highlighted, as well as the code that is executed between the uses and reuses that are represented by these arrows. From the visualization (see Fig. 2(b)), it can be concluded that loop fusion will remove these long-distance reuses, removing 5.4% of the capacity misses.

Loop Tiling. When selecting the cluster displayed in Figure 4, one sees that 75% of all long distance reuses are caused by this cluster. Figure 3 shows that between use and reuse, most of the memory references in the outer loop are executed. Therefore, the reuse occurs between different iterations of this outer loop, and the reuses can be brought together by tiling the loop.

5 Conclusions and Future Work

We have presented locality metrics (reuse distance, reference pairs and associated basic block vectors) that capture the necessary information to optimize the locality of programs. These metrics are measured by instrumenting the program and visualized after executing the program. The visualization is unique in the aspect of clustering the source code locations that lead to cache misses, so that each cluster needs a single program transformation to remove these cache misses. This enables the programmer to find the necessary program transformations to improve locality in a time-scale of minutes, whereas previously this could take days. The visualization proofs effective, since the three SPEC2000 programs with the largest memory bottlenecks could be optimized by it, leading to a three-fold speedup on average on a number of different platforms.

In the future, we would like to extend the presented work in a number of ways. First, we are thinking about using program slicing methods to automatically find independent code that is executed between use and reuse. Secondly, the possibility for predicting reuse distance distributions of long program runs, based on a short instrumented execution with small inputs[6], will be investigated. Thirdly, we are thinking about cluster analysis methods to guide data layout optimizations to improve spatial locality.

The original and optimized programs from the experiments are available at http://www.elis.UGent.be/~kbeyls. We plan to make the rdvis tool and the instrumentation patch for GCC available at the same place, after GCC 4.0 is released.

References

1. E. Berg and E. Hagersten. Sip: Performance tuning through source code interdependence. In *Euro-Par'02*, volume 2400 of *LNCS*, pages 177–186, 2002.
2. K. Beyls and E. D'Hollander. Platform-independent cache optimization by pinpointing low-locality reuse. In *ICCS 2004: 4th International Conference on computational science*, volume 3038 of *LNCS*, pages 448–455, 2004.
3. K. Beyls and E. H. D'Hollander. Reuse distance as a metric for cache behavior. In *Proceedings of PDCS'01*, pages 617–662, Aug 2001.
4. R. Bosch and C. S. et al. Rivet: A flexible environment for computer systems visualization. *Computer Graphics-US*, 34(1):68–73, Feb. 2000.
5. J. F. Cantin and M. D. Hill. Cache performance for selected SPEC CPU2000 benchmarks. *Computer Architecture News (CAN)*, September 2001.
6. C. Ding and Y. Zhong. Predicting whole-program locality through reuse distance analysis. In *PLDI'03*, ACM SIGPLAN Notices, 2003.
7. S. Ghosh. *Cache Miss Equations: Compiler Analysis Framework for Tuning Memory Behaviour*. PhD thesis, Princeton University, November 1999.
8. A. Goldberg and J. Hennessy. Mtool: An integrated system for performance debugging shared memory multiprocessor applications. *IEEE Transactions on Parallel and Distributed Systems*, 4(1):28–40, 1993.
9. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.
10. A. R. Lebeck and D. A. Wood. Cache profiling and the SPEC benchmarks: A case study. *IEEE Computer*, 27(10):15–26, 1994.
11. M. Martonosi, A. Gupta, and T. Anderson. Tuning memory performance in sequential and parallel programs. *IEEE Computer*, April 1995.
12. K. S. McKinley, S. Carr, and C.-W. Tseng. Improving data locality with loop transformations. *ACM Transactions on Programming Languages and Systems*, 18(4):424–453, July 1996.
13. J. Mellor-Crummey and R. F. et al. HPCView: a tool for top-down analysis of node performance. *The Journal of Supercomputing*, 23:81–104, 2002.
14. T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. *ASPLOS, SIGPLAN Notices*, 37(10):45–57, 2002.
15. E. Vanderdeijl, O. Temam, E. Granston, and G. Kanbier. The cache visualization tool. *IEEE Computer*, 30(7):71, 1997.
16. J. Weidendorfer, M. Kowarschik, and C. Trinitis. A tool suite for simulation based analysis of memory access behavior. In *ICCS 2004: 4th International Conference on computational science*, volume 3038 of *LNCS*, pages 440–447, 2004.
17. Y. Yu, K. Beyls, and E. D'Hollander. Visualizing the impact of cache on the program execution. In *Information Visualization 2001*, pages 336–341, 2001.

CacheIn: A Toolset for Comprehensive Cache Inspection

Jie Tao and Wolfgang Karl

Institut für Rechnerentwurf und Fehlertoleranz,
Universität Karlsruhe (TH), 76128 Karlsruhe, Germany
{tao, karl}@ira.uka.de

Abstract. Programmers usually rely on cache performance data to optimize their applications towards high runtime cache hit ratio. In this paper, we introduce a software toolset CacheIn, which uses simulation and monitoring to collect comprehensive cache performance data. CacheIn consists of a cache simulator for modeling the cache activities, a cache monitor for gathering different kind of information, and a multilayer software infrastructure for processing the raw monitoring data towards statistical, high-level representations, like histograms and summarized numbers. CacheIn exhibits both the details of traditional software mechanisms and the feasibility of performance counters. Based on a code instrumentor, we have verified CacheIn using standard benchmarks. Initial experimental results show its full functionality in terms of providing accurate, comprehensive, and corse-grained performance data.

1 Introduction

Cache locality optimization is regarded as a critical issue for achieving high performance. A prerequisite for such optimization is performance data that shows the cache access behavior of applications. Currently, computer systems rely on either software profiling or hardware counters to acquire this information . However, both approaches can provide only limited performance data; and information about important performance metrics, like false sharing, cache line invalidation, cache line replacement, and access pattern, is missing.

We developed CacheIn for acquiring sufficient information that allows comprehensive analysis and optimization. For feasibility, we use a cache simulator to model various cache organizations and different caching policies. This allows to apply CacheIn to study caches on different target machines. For comprehensive data, we implemented a software model of a cache monitor capable of observing the memory traffic on all levels of the memory hierarchy and collecting detailed information about the cache access behavior. This monitoring facility can be configured to a variety of working modes at the runtime and provide different information needed for understanding the various access pattern of applications and for the selection of appropriate optimization techniques. Additionally, in order to avoid delivering fine-grained, low-level performance data, a multilayer software infrastructure has been developed for transforming the original monitoring information into a high level abstraction with respect to data structures. This includes both APIs for address transformation and interfaces for convenient

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 174–181, 2005.
© Springer-Verlag Berlin Heidelberg 2005

access of the performance information. Based on the software infrastructure, CacheIn provides, for example:

- Access histograms on individual location or the whole memory hierarchy. They record the access distribution to the complete working set at granularity of cache lines. This gives the user a global overview of the memory accesses allowing an easy detection of access hotspots.
- Statistics on single events, like cache misses and total references to a specific memory region or performed inside an individual iteration, loop, or function; number of cache line replacements within an array; and number of first references with respect to memory regions, arrays, or code regions. This allows to find the sources causing cache miss and inefficiency.
- Profile of access addresses. Based on this information, accesses at close intervals can be grouped into the same cache line, thus reducing cache misses caused by first references and also increasing spatial reuse of the cached data.
- Sequence of cache events which records the frequency of replacements and numbers of hits between a first reference and a replacement performed on the same memory line. This helps to hold frequently reused data in the cache prohibiting frequently replacement and reloading.
- Additional information for multiprocessor systems, e.g. information about false sharing. This helps to reduce the number of cache line invalidations and thereby improve the cache efficiency.

CacheIn requires memory references as input for its cache simulator. This can be e.g. extracted from debugging information, or acquired from profiling tools and compilers. For this, CacheIn provides a simple interface to theses systems. For example, to verify CacheIn's functionality, we have applied a code instrumentor to provide memory references.

The remainder of this paper is organized as following. Section 2 introduces several related works in acquiring cache performance data. This is followed by a detailed description of this toolset in Section 3, including the cache simulator, the cache monitor, and the software infrastructure. In Section 4 some initial experimental results are illustrated. The paper concludes in Section 5 with a short summary.

2 Related Work

As locality tuning requires information about memory accesses, various approaches have been developed for collecting performance data with respect to the memory system. These approaches can be roughly divided into two categories: hardware supported and simulation/profiling based.

On the area of hardware, modern processors provide performance counters for recording important information about the runtime execution. The UltraSPARC IIi Architecture [6] provides two registers to count up to 20 events like cache misses, cache stall cycles, floating-point operations, branch mispredictions, and CPU cycles. The IBM POWER2 [7] has 5 performance counters enabling the concurrent monitoring of five events, such as the number of executed instructions, elapsed cycles, and utilization

of the various execution elements. Intel's Pentium4 [2] supplies even 18 performance counters to allow the collection of more information, like that about specific instructions and pipeline conflict.

For simulation/profiling based schemes, well-known examples are SIP [1], MemSpy [4], and Cachegrind [9]. SIP (Source Interdependence Profiler) is a profiling tool that provides information about cache usage and cache miss ratios. It uses SimICS [3], a full-system simulator, to run the applications for collecting cache behavior data. MemSpy is a performance monitoring tool designed for helping programmers to discern memory bottlenecks. It uses cache simulation to gather detailed memory statistics and then shows frequency and reason of cache misses. Cachegrind is a cache-miss profiler that performs cache simulations and records cache performance metrics including L1 instruction cache reads and misses, L1 data cache accesses and misses, and L2 unified cache accesses and misses. The input for Cachegrind is extracted from the debugging information.

Overall, the hardware counters provide only limited performance data and details are missing; programmers often have to do hard work in analyzing the applications in order to understand the code and the data structure. For simulation/profiling based approaches, performance data provided by existing implementations is also restricted to numbers of specific events and this information does not suffice for a full understanding of the cache access behavior. We therefore design a toolset in order to provide the programmers with detailed, understandable, and easy-to-use performance data which is necessary for efficient code and data optimization.

3 The Approach

The goal of this work is to deliver comprehensive cache performance data that not only shows the various aspects of cache behavior but also is easy to use, e.g. in the form of statistics and at high-level in terms of data structures. We achieve these features with a software framework consisting of a flexible cache simulator, a generic cache monitor, and a multilayer software infrastructure. A high-level overview of this framework is depicted in Figure 1.

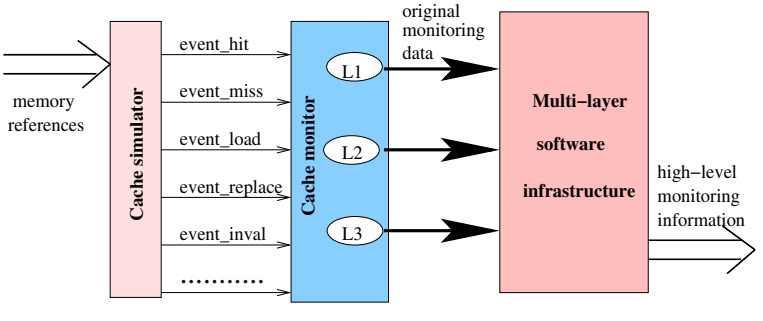


Fig. 1. CacheIn software infrastructure on the top level

3.1 Cache Simulation

We implemented a flexible cache simulator to model the operations within the caches. Its input are memory references which can be provided by debuggers, simulators, or compilers. Based on the access address and transaction type, it simulates the process of cache searching. In comparison with the similar work in this area, e.g. the cache simulator in Cachegrind [9], our simulator distinguishes in its flexibility, feasibility, and comprehensiveness.

First, it models a multilevel cache hierarchy and allows each cache to be organized as either write-through or write-back depending on the target architectures. All relevant parameters including cache size, cache line size, and associativity (from directly mapped to fully associative) can be specified by the user.

In addition to simulating the caches themselves, the cache simulator models a set of cache coherence protocols. This includes the hardware-based MESI protocol, several relaxed consistency schemes, and an optimal, false-sharing free model using a perfect oracle to predict required sharing. This property of the cache simulator can be used to understand the invalidation behavior and further to develop adaptive mechanisms for optimization.

The output of the cache simulator are various events that can be directly delivered to the cache monitor. As shown in Figure 1, the cache simulator generates a set of different type of events, including hit, miss, load, replacement, and invalidation. Besides the type, an event also holds parameters, such as access address, cache level, and transaction type, which are needed for the cache monitor to achieve its functionality.

3.2 Cache Monitoring

To acquire performance data capable of reflecting the various aspects of cache accesses, a cache monitor is needed. For flexibility, modularity, and unified interfaces and construct, several specific requirements must be considered in the monitor design: 1) each cache level needs an individual monitor and all monitors have the same structure; 2) monitors can be configured to a variety of working modes depending on the user demand; 3) a monitor must have the capability of event filtering, aggregating, and preprocessing for avoiding to deliver information of unessential details.

Following these requirements, we designed a novel monitoring concept and implemented a generic, multi-functional monitoring component which can be combined to any location of the cache hierarchy. This cache monitor is comprised of an input interface, an output interface, and a flexible analysis module. The former is an interface to the cache simulator, while the output interface allows the data to be delivered to the software infrastructure for further processing.

As the primary component of the cache monitor and also one of the most distinguished part of this work, the analysis module is responsible for event handling. It has mechanisms for bypassing events of no user interest. It also provides a dynamic granularity control that allows to monitor single words for e.g. detecting false sharing and that enables the aggregation of neighboring events in case of access histograms where fine granularity is not required. For this granularity control it deploys a counter array to transiently store the processed events, and a ring buffer to hold the later generated monitoring data.

Another feature of the analysis module is its various functions. It can be specified to work in a set of modes, either triggering user-defined events and counting their occurrence or working in a histogram-driven mode where it captures all related events for generating a certain type of histograms, like memory access histogram recording all accesses to the complete working set, replacement histogram that stores each cache line replacement in chronological order, or histogram of access addresses providing a series of access targets.

3.3 Software Infrastructure

The monitoring data delivered by the cache monitor is still event-based, fine-grained, and hence not suitable for directly providing to users or performance tools that usually need high-level representations in the form of access histograms and summarized access numbers. For this, a software infrastructure has been developed.

As illustrated in Figure 2, the software infrastructure contains several layers, each processing a step further of the monitoring information. As mentioned, the cache monitor stores the monitoring data in a ring buffer. From there, data is sorted by the Control Component and stored into a histogram chain that is ordered by the addresses of corresponding memory blocks with cache line size, so called memory lines.

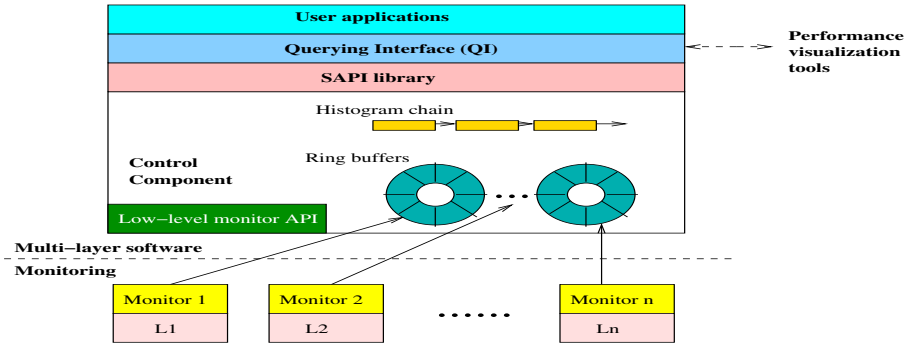


Fig. 2. Software infrastructure for data processing

In addition, the Control Component also combines the monitoring data from different monitors and probably different processors (on a multiprocessor system). On top of this component, the SAPI (Standardized API) library further processes the monitoring data into statistical forms such as total number of single events and access histograms. Finally, the Querying Interface (QI) maps the virtual addresses into data structures using the context table provided by some compilers or using the debugging information. Also from this component, the final high-level data abstraction is delivered to performance tools and applications for visualization or performance analysis.

As the primary data processing component, SAPI provides a set of functions for generating both statistical numbers on individual events and access histograms recording the occurrence of single events over the complete working set. In addition, SAPI provides functions for analyzing the access addresses and invalidation operations. These

functions can be used to provide address groups and to create invalidation sequences, which are needed in address grouping and false sharing detection.

4 Verification Based on a Code Instrumentor

In order to verify the functionality of CacheIn, we have deployed Doctor [5], a code augmenter, to deliver memory references. Doctor is originally developed as a part of Augmint [5], a multiprocessor simulation toolkit for Intel x86 architectures. Doctor is specially developed for augmenting the assembly code with instrumentation instructions that generate memory access events. For every memory reference, Doctor inserts code to pass the accessed address and its size to the simulation subsystem of Augmint. For this work, we have slightly modified Doctor in order to build an interface to the cache simulator of CacheIn.

Actually, CacheIn is capable of supplying a variety of performance data, like statistics on single events, individual access histogram on a single cache level, combined histogram on the whole memory hierarchy, and operation histogram showing ordered events. We choose two examples to show its basic function: complete access histogram and statistics on false sharing.

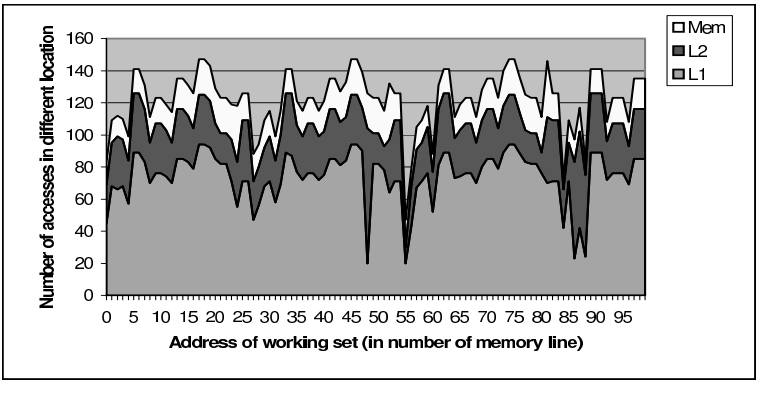


Fig. 3. Combined memory access histogram of WATER

A complete access histogram consists combined data for all monitors on a cache hierarchy. Figure 3 illustrates a sample histogram with WATER, an application for evaluating water molecule systems. For this, we simulated the cache behavior of WATER on a system with two caches. For the concrete diagram in Figure 3, the x-axis presents the first 100 memory lines of the complete working set and the y-axis presents the number of accesses performed on each memory line. These accesses can be either L1 hits, L2 hits, or have to be performed in the main memory. As can be seen, most memory references can be found in the caches, however, for this concrete example there also exists memory regions with a high access rate of the main memory. Hence, the access histogram enables a direct observation of distinct memory access behavior within a

single code. This allows to find access hotspots, forming the first step towards cache optimization.

Our second example addresses multiprocessor systems with shared memory. On such a system, cache line invalidation is an additional critical issue causing cache misses. Such invalidations, however, could be unnecessary. A specific case is false sharing, where a cache line on a processor is invalidated because another processor has modified a word of the same data copy; but the processor needs a different word within the cache block. In this case, it is not necessary to invalidate the cache line.

For supporting the optimization with respect to false sharing, the cache monitor of CacheIn provides event profile that records the histogram of memory operations in a serial order. This allows its API to compare the target of a shared write with all following shared reads thus to detect false sharing. Table 1 shows the statistics reported by the cache monitor API. This result was acquired by simulating several applications on a 32 processor system, including a Fast Fourier Transformation (FFT) code, an LU-decomposition code for dense matrices (LU), an integer radix sort (RADIX), the OCEAN code for simulation of large scale ocean movements, and the WATER code. All these applications are chosen from the SPLASH-II Benchmarks suite [8].

We have measured the number of false sharing for four different cache coherence protocols. MESI is a common used scheme for hardware-based shared memory machines. This scheme performs cache line invalidation by each write operation to shared data. FULL is a kind of release consistency model usually deployed on systems with distributed shared memory. This scheme performs whole cache invalidation at each synchronization event like lock and barrier. OPT is an optimal scheme that invalidates a cache line by a read operation and only when the accessed cache line has been modified. SCOPE is an optimized version of FULL, where only the cache lines holding remote data are invalidated rather than the complete cache. In principle, OPT should performs better than MESI, MESI better than FULL, and SCOPE better than FULL.

Table 1 depicts the absolute number of total invalidations and false sharing of them. It can be observed that applications vary in this behavior. For FFT, 40% of the invalidations with MESI are false sharing, 50% with OPT, and even 78% with SCOPE. For the FULL scheme, it is senseless to compute this proportion because most invalidations are performed on invalid cache lines. However, more false sharing can be observed with this protocol. LU performs better than FFT with a slight percentage of false sharing, e.g. 3% with MESI. RADIX reports the highest false sharing (65% with MESI) and the other two applications perform better than RADIX and FFT.

Table 1. Number of false sharing with different cache coherence schemes

	MESI		FULL		OPT		SCOPE	
	invalid.	false share.	invalid.	false share.	invalid.	false share.	invalid.	false share.
FFT	2670	1057	18517	2626	992	496	2110	1647
LU	16185	487	304137	4728	3392	24	106380	3886
RADIX	7274	4716	63857	5488	654	160	15937	4774
OCEAN	17990	2875	253808	4405	8411	2538	133490	2748
WATER	9302	2704	70662	4486	2920	2557	20590	2378

Overall, the experimental results indicate that for most applications optimization with false sharing is necessary in order to alleviate cache misses. For this, CacheIn provides required data for analysis.

5 Conclusions

This paper introduces a software framework developed for acquiring comprehensive, accurate, and detailed performance data about the cache access behavior. CacheIn is comprised of a cache simulator for modeling the cache activity, a cache monitor for gathering original performance data, and a multilayer software infrastructure for creating statistical monitoring information in the form of histograms and summarized number. The specific feature of CacheIn lies in its feasibility, flexibility, and the capability of providing comprehensive performance data with various details. This has been proven by the verification based on a code augmenter.

References

1. E. Berg and E. Hagersten. SIP: Performance Tuning through Source Code Interdependence. In *Proceedings of the 8th International Euro-Par Conference*, pages 177–186, August 2002.
2. Intel Corporation. *IA-32 Intel Architecture Software Developer's Manual*, volume 1–3. 2004. available at Intel's developer website.
3. P. S. Magnusson and B. Werner. Efficient Memory Simulation in SimICS. In *Proceedings of the 8th Annual Simulation Symposium*, Phoenix, Arizona, USA, April 1995.
4. M. Martonosi, A. Gupta, and T. Anderson. Tuning Memory Performance of Sequential and Parallel Programs. *Computer*, 28(4):32–40, April 1995.
5. A-T. Nguyen, M. Michael, A. Sharma, and J. Torrellas. The augmint multiprocessor simulation toolkit for intel x86 architectures. In *Proceedings of 1996 International Conference on Computer Design*, October 1996.
6. Sun Microsystems. *UltraSPARC IIi User's Manual*. October 1997. available at http://www.sun.com/processors/documentation.html.
7. E. Welbon and et al. The POWER2 Performance Monitor. *IBM Journal of Research and Development*, 38(5), 1994.
8. S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 24–36, June 1995.
9. WWW. Cachegrind: a cache-miss profiler. available at http://developer.kde.org/šewardj/ docs-2.2.0/cg_main.html#cg-top.

Optimization-Oriented Visualization of Cache Access Behavior

Jie Tao and Wolfgang Karl

Institut für Rechnerentwurf und Fehlertoleranz,
Universität Karlsruhe (TH), 76128 Karlsruhe, Germany
{tao, karl}@ira.uka.de

Abstract. Cache performance is regarded as a critical issue that significantly influences the overall performance of a computer system. Hence, optimization with respect to cache behavior forms an important topic in both research and commercial areas. This work aims at helping the user in the task of cache behavior analysis with a comprehensive visualization tool. The resulted cache visualizer is capable of presenting the various aspects of cache accesses, and shows the performance data in a high-level, user-readable way. More specially, it directly leads the user to the data structures where excessive cache misses occur. It then further depicts the reason of theses misses. In addition, it helps the user to decide important parameters concerning some optimization techniques, e.g. padding stride and size of cache blocks. Examples of initial optimization based on the visualizer have proven the feasibility of this visualization tool.

1 Introduction

Cache memories have been long introduced in the computer system to relax the widening gap between the memory and processor speed. However, due to poor cache locality, applications still face performance problems caused by long memory access latency. Locality optimization is therefore becoming increasingly important in current systems.

As a result, various approaches have been proposed for enhancing the cache efficiency. These approaches can be roughly divided into two categories: compiler based data/code restructuring and explicit optimization within source codes. The former is done transparently at compile time through code transformation [4], data structure padding [6], or both of them [1], based on heuristic analysis and mathematical frameworks; while the latter [2, 5] is achieved by manually rewriting the program code using the same techniques as compilers.

Since compiler-based approaches have to integrate both analysis and optimization into the complex structures of modern compilers, programmers prefer to using explicit optimization that can be directly performed in the source code. This, however, needs a full understanding of applications' runtime access patterns with respect to the cache behavior, which can be quite tedious due to the complex data structure of applications and the memory hierarchy.

Therefore, a set of cache visualization tools have been developed in order to help the user in this task. These tools, however, either are not optimization-oriented or provide

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 182–190, 2005.
© Springer-Verlag Berlin Heidelberg 2005

only limited information, and hence can not really indicate what and where the cache problems are and how to tackle these problems.

We designed and implemented a new cache visualizer with the goal of showing the various aspects of cache accesses and behavior. It displays the performance data in a high-level, user-readable way and can lead the user to the data structures where excessive cache misses occur. It then depicts the reason of theses misses and helps the user to deduce reasonable values for parameters used in different optimization schemes. More concretely, this cache visualizer can:

- depict the bottlenecks where excessive references are performed on the main memory. This view allows both an overall observation of the complete working set and a focus on specific memory regions or code segments, like individual functions, loops, and iterations.
- show the different access behavior in various program phases using 3D views. This allows the user to have a deeper understanding of the cache performance and further to perform more detailed optimization on a per-phase basis.
- exhibit the sources of cache misses in terms of data structures and give information that allows the user to decide parameters for array and code transformation.
- illustrate the impact of optimizations, which helps the user to deploy appropriate optimization techniques or to improve the existing schemes.
- show the behavior of cache line invalidation and false sharing for shared memory multiprocessor systems. This allows additional performance gain that can be introduced by optimizing cache coherence protocols.

Overall, we have developed a comprehensive cache visualization tool. It focuses on the presentation of cache problems in a user-understandable way. Users benefit from it in easy detection of access hotspots and deduction of corresponding optimization schemes.

The remainder of this paper is organized as follows. Section 2 gives a short introduction of related work in this research area. This is followed by a brief description of the design requirements for a comprehensive cache visualizer in Section 3. Then in Section 4 the resulted cache visualizer is described in detail. The paper concludes in Section 5 with a summary and some future directions.

2 Related Work

In the field of cache visualization, several tools have been developed. The cache visualizer described in [9] is an example. Similar to our cache visualizer, this tool aims at helping the user to optimize the source code towards less cache misses. For this, it provides two graphical views showing the cache access behavior. The first view is called "cache miss pattern" and is used to illustrate all memory references, each presented with a colored pixel that depicts whether an access is a cache hit or a kind of cache miss. The other view shows the reuse distance which is defined as the number of distinct memory lines fetched between two memory accesses to the same memory line. Again all memory references are illustrated within a single diagram, using different colors to represent

the reuse distance for each access. This view aims to show the feature of capacity misses for fully associative caches, helping the user thereby to detect an appropriate cache size for the applications. However, both views are low-level and not possible of leading the user to form a picture about how the cache behaves at the runtime.

CVT [8] is another cache visualization tool with the focus on showing the behavior of interference misses. It provides two windows, one for showing statistics on cache misses in terms of arrays, and the other, its main window, is a grid illustrating the content of cache lines holding the elements of a working array. The goal of the latter is to allow the detection of elements in different arrays that are mapped to the same cache line causing conflicts. However, this approach is restricted to directly mapped caches, which do not represent the technology of modern processors.

The Intel VTune Performance Analyzer [3] is regarded as a useful tool for performance analysis. It provides several views, like "Sampling" and "Call Graph", to help identify bottlenecks. For cache performance it only shows number of cache misses. Even though this information can be viewed by process, thread, module, function, or instruction address, it is not sufficient for optimizations with respect to cache behavior.

Hence, we implemented a comprehensive, optimization-oriented cache visualizer capable of showing the different aspects of runtime cache behavior. We use high-level, user-understandable graphical views to allow not only the presentation of access hotspots, miss reason, and impact of optimization, but also the deduction of important parameters for code and data transformation. In the following, we first discuss the design requirements for a comprehensive cache visualization tool and then describe this cache visualizer in detail.

3 Design Challenges

A visualization tool, which targets at helping the user to understand the behavior of cache operations and the runtime activity of the code, has to fulfill a set of user requirements. First, it must provide users with both an overview of the complete application access behavior, as well as views of the detailed behavior within specific execution regions, such as a single loop nest, individual function, or specific data segments. This allows the user to begin with a high-level behavioral view, and then to successively refine the focus on hotspots and bottlenecks.

An optimization-oriented tool also must map the gathered behavioral data back to data structures within the source code. Since memory addresses are not visible within an application program, this correlation is necessary for the user to directly perform the optimization on the arrays within the source code.

In addition, access behavior often changes in different phases of an application, and the visualization of cumulative behavior alone may actually hide these differences. A tool therefore needs to show access patterns in different program phases, which may either be explicitly specified or may be identified by barriers, locks, or other synchronization mechanisms. This allows the user to detect the specific locality requirements of each execution phase, and hence enables more comprehensive optimization strategies.

Finally, a cache visualization tool must be not only capable of showing access bottlenecks but also able to give the reason of those cache misses. Only this enables the

detection of exact target to optimize and the selection of appropriate optimization techniques. However, most cache visualization tools fail to do this.

4 The Cache Visualizer

According to the design requirements described above, we implemented an optimization-oriented cache visualizer. In this section, a toolset for acquiring cache performance data is first introduced and then we go into the details of related issues with this visualization tool.

4.1 Source of Performance Data

We deploy CacheIn [7], a monitoring and simulation framework, to acquire performance data. CacheIn, at the top level, is comprised of a cache simulator, a cache monitor, and a software infrastructure. The former takes memory references as input and then models the complete process of data searching in the whole memory hierarchy with multilevel caches. According to the search result, it generates events for the cache monitor, e.g. hit/miss, load, replacement, and invalidation events.

The cache monitor defines a generic monitoring facility capable of being connected to any location of the memory system. It provides mechanisms to filter and analyze the events and then generates low-level monitoring data. More specially, it offers a set of working modes allowing the acquisition of different kind of cache performance data. This data is further delivered to the software infrastructure which produces higher level information in statistical form like histograms and total numbers. This forms the basis for the cache visualizer to exhibit the cache accesses in different aspects.

4.2 Visualization Infrastructure

The cache visualizer uses a client-server infrastructure to coordinate the interoperation between the target system, the data generator (CacheIn), the users, and the visualization facilities.

As the core of this infrastructure, the server receives the users' view selections and the monitoring data. According to the views, it then processes and prepares the data into required form and initiates the visualizer to generate the specified graphical view. The server is capable of handling user requests concurrently to processing the delivered monitoring data. Built on top of the server, the visualizer shows the required information in well understandable representations.

In addition, a graphical user interface is deployed for execution control, view selection, and on-line steering. This GUI consists of two different parts: the first part allows the specification of commands and parameters for executing an application, while the second part allows the user to select a graphical view of interest. The control information for execution is delivered to the target system and the visualization requests are transfered to the server. This means that the machine running the application under observation can be run separately from the machine performing the actual visualization allowing the use of special rendering equipment wherever it is available.

4.3 Available Views

The cache visualizer provides a set of views to show the various aspects of the cache behavior. This includes a few initial views based on virtual addresses and several higher level representations concerning data structures and dynamic accessing patterns. Due to restricted space, only a few sample views will be depicted in the following after the view description.

Access Histogram. First, a 2D diagram shows the overall access histogram covering the complete working set. It uses different colors to present the locations (cache or memory) of all memory references to a single memory line or array element. This allows hence to highlight the regions where large amount of accesses to the main memory are performed.

Phase Information. A 3D view shows a set of access histograms, each for a single program phase that can be identified by iterations and synchronization operations, or for an individual code region that is specified by functions and routines. Corresponding to each phase, a 2D view, like described above, is combined. The 3D view hence enables a detailed insight into the access pattern of different phases and directs a per-phase optimization capable of handling the different access behavior existing in different program phases and functions. For example, the first program phase usually handles the initialization of working arrays and hence varies from other phases in access behavior. Other phases can also behave differently. A specific optimization with respect to individual phase can therefore further introduce performance gains.

Cache Miss Reason. The previous views allow the user to find access bottlenecks. This is further supported by another 2D view which is used to exhibit the reasons causing the bottlenecks. For each critical memory region it shows the proportion of three main miss sources: first reference, replacement, and cache line invalidation. This aids the user to choose an adequate technique for tackling the bottlenecks. The following two views supply more information for this.

Address Series: This view shows the repeated series of access addresses in different lengths for a user specified start address. According to the frequency of each series users can decide which access addresses should be grouped and placed chronologically in the main memory. Such optimization can significantly reduce cache misses because loading the first address in a series also bring the next required data into the cache.

Memory Operation Histogram: This view shows the operations performed on a specified cache area, normally the critical region where inefficiency occurs. The operations include loads, hits, and replacements. Loads represent the loading of a line into the cache, hits imply cache reuse, and replacements indicate the data exchange. Users can specify a histogram either for showing all kinds of events or a single. The change of status with each cache line within the displayed area is highlighted in different color, allowing to detect those lines with heavy traffic – data is frequently loaded into and removed from. Actually, these lines shall be held in the cache during the whole execution. In addition, this view also presents the overmapping of reused data, aiding the user to

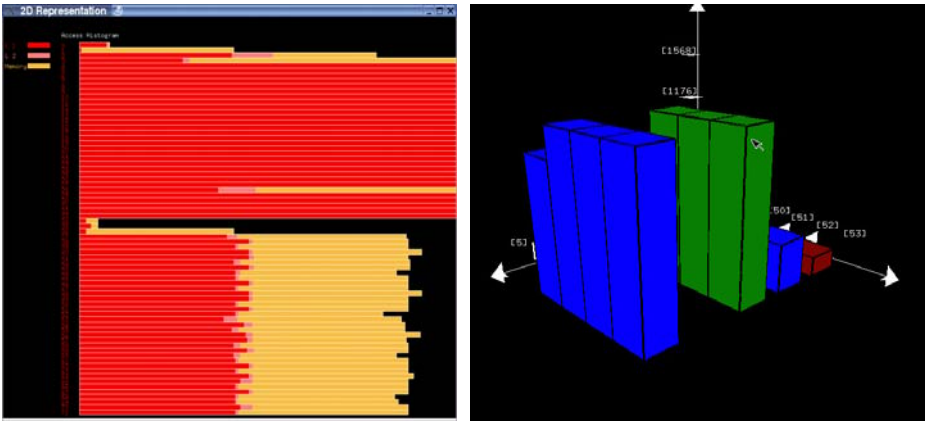


Fig. 1. Sample views: 2D overview (left) and 3D phase (right)

decide some parameters for optimization such as padding stride, size of blocking, and depth of code permutation.

False Sharing. This view shows the operations to each word within a specified cache line. Together with the accessed word, the type of operation and the processor performing this operation are also presented. Relying on the operations presented one after another, it is easy to find false sharing where a cache line is first written by a processor and then read by another processor but targeting a different word within this cache line. In this case, the corresponding line in the cache of the second processor has not to be invalidated. Therefore, this view helps to reduce cache line invalidations and thereby also cache misses.

Figure 1 depicts a 2D Access Histogram and a 3D Phase Information. As mentioned, the Access Histogram gives an overview of the access distribution on the memory system. This can be based on virtual memory lines or data structures. The concrete diagram in Figure 1 is an example of the former and presents an access histogram on a memory hierarchy with two caches. Corresponding to each virtual memory line, which is labeled left to the diagram, three colored bars show the number of accesses performed at each location. Access hotspots, where a large amount of accesses are fulfilled on the main memory, like the last several lines, can be highlighted in this way. Similarly, the 3D Phase Information also shows the access histograms, but in a per-phase fashion with each phase presenting the access behavior within a single program phase or an individual function. Hence, it contains a set of 2D histogram views. For the concrete diagram shown in Figure 1, each 2D view displays the miss distribution to the chosen data structure used in a single routine. This allows to detect concrete cache problems within a specific code region, which probably could not be exhibited by a global overview. By zooming into a single phase, the corresponding 2D diagram is shown in more detail, allowing a deeper examination of the problems in this phase. Hence, the 3D view enables a more comprehensive cache optimization.

Fig. 2. Sample view: Memory Operation Histogram

Figure 2 illustrates a sample Memory Operation Histogram created when executing the Fast Fourier Transformation (FFT) code on a system with a 4-way, 8k L1 cache and a 4-way, 128K L2 cache. This view contains three components for showing the mapping of main data structures in the cache, the runtime cache actions, and the memory location of the data structures respectively. The first component, as depicted in the left of Figure 2, gives the location of array blocks in the cache. For example, the first block of array X is mapped to the second set of the first level cache. The memory addresses of the arrays are presented in the right component. As the primary part of this view, the cache actions are shown in the middle. It depicts the changes of line states and contents in the user-specified cache sets. The status of a cache line can be replaced, hit, loaded, or empty, and is identified with colors. The visualization is synchronized with the actual cache operation, i.e. it shows the changes step by step according to the runtime data accesses. Users can also observe the complete operation series within a single set by clicking at the set area. Hence, this view helps to find overmapping between array blocks and to deduce optimization parameters, such as the padding stride, for eliminating these mapping conflicts.

4.4 Optimization: A Case Study

We use the LU code as an easy-to-follow example to depict how the visualization tool helps to improve the cache behavior. LU factors a dense matrix into the product of a lower triangular and an upper triangular matrix. The dense matrix is divided into blocks to exploit temporal locality on submatrix elements. We executed this program and simulated the cache behavior with CacheIn. The first CachIn reports show high cache misses.

According to the visualizer's presentation, we found that these cache misses are mainly caused by accessing the matrix being decomposed. Besides this, the visualizer

shows that theses misses are caused by cache line replacement on the matrix itself. Based on the Memory Operation Histogram view, we then found that several matrix blocks of LU, which are needed for a single computation, are mapped into the same cache sets at the runtime leading to replacement. According to this detection, we optimized the LU code with a better data allocation to prohibit these overmappings. We executed LU and simulated the cache again but with the optimized code. An improvement with cache locality was achieved.

5 Conclusions

In this paper, we introduce a visualization tool for presenting the cache access behavior. In order to overcome the deficits of existing visualizers in supporting locality optimization, this cache visualizer is specially designed. Unlike the conventional visualization tools which show "what happened", our toolset shows the problem, the reason, and the solution, e.g. "where the bottleneck lies", "what results this bottleneck", and "how to correct it". For this, it deploys a variety of graphical views with different dimension, granularity, and data basis. A sample optimization based on this tool has proven its feasibility and effectiveness.

In the next step of this research work, we will use the visualization platform to further study the impact of cache locality optimizations with complicated, large applications. Then the optimization will be moved to realistic applications and real hardware systems. This step also brings potential extensions to the established cache visualizer.

References

1. C. Ding and K. Kennedy. Improving Cache Performance in Dynamic Applications through Data and Computation Reorganization at Run Time. *ACM SIGPLAN Notices*, 34(5):229–241, May 1999.
2. C. C. Douglas, J. Hu, M. Kowarschik, U. Rüde, and C. Weiss. Cache Optimization for Structured and Unstructured Grid Multigrid. *Electronic Transaction on Numerical Analysis*, 10:21–40, 2000.
3. Intel Corporation. Intel VTune Performance Analyzer. available at http://www.cts.com.au/vt.html.
4. N. Megiddo and V. Sarkar. Optimal Weighted Loop Fusion for Parallel Programs. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 282–291, New York, June 1997.
5. J. Park, M. Penner, and V. Prasanna. Optimizing Graph Algorithms for Improved Cache Performance. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, pages 32–33, April 2002.
6. G. Rivera and C. W. Tseng. Data Transformations for Eliminating Conflict Misses. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 38–49, Montreal, Canada, June 1998.
7. J. Tao and W. Karl. CacheIn: A Toolset for Comprehensive Cache Inspection. In *Proceedings of ICCS 2005*, May 2005.

8. E. van der Deijl, G. Kanbier, O. Temam, and E. Granston. A cache visualization tool. *Computer*, 30(7):71–78, July 1997.
9. Y. Yu, K. Beyls, and E. D'Hollander. Visualizing the Impact of the Cache on Program Execution. In *Proceedings of the fifth International Conference on Information Visualization*, pages 336–341, July 2001.

Collecting and Exploiting Cache-Reuse Metrics

Josef Weidendorfer and Carsten Trinitis

Technische Universität München, Germany
{weidendo, trinitic}@cs.tum.edu

Abstract. The increasing gap of processor and main memory performance underlines the need for cache-optimizations, especially on memory-intensive applications. Tools which are able to localize code regions with high cache miss ratio seem to be appropriate for access optimizations. However, a programmer often does not know what to do with the collected information. We try to improve this situation by providing cache reuse metrics which are supposed to give more precise hints on how to optimize memory access behavior. We enhanced the cache simulator Callgrind to give metrics on temporal and spatial cache utilization for a given memory block, relating this information to the code line where the block was loaded into cache. We show what is needed for hardware-supported measurement for such metrics, and give example code where the collected information directly points to optimization possibilities.

Keywords: Cache Reuse Metrics, Profiling, Cache Simulation.

1 Introduction

A well-known problem for memory-intensive applications is the so-called memory wall: while CPU performance still increases according to Moore's Law at a rate of around 70 % per year, main memory performance improves only by around 7 % per year, leading to an enlarging gap [15]: on modern processors, main memory latency is already around 200-300 processor cycles, causing a large speed impact. This bottleneck will become even more obvious with multi core, multi threaded processors, which quite easily can lead to a fourfold bandwidth requirement.

To reduce this problem, two strategies can be applied: the first is cache optimization by reordering memory accesses or rearranging data-layout, to improve temporal and spatial locality of an application [8]. This approach aims at optimal cache utilization. The second way is prefetching, i.e. pre-loading data into cache which is needed in the future, and thus trying to exploit the available transfer rate to main memory. For both techniques, to locate the code positions where according changes are required, profiling tools are used. These tools relate cache events happening to the code executed at these points in time. Quite simply to implement, and provided in most contemporary processors, are hardware performance counters. Either the application code is instrumented around interesting code regions for gathering exact counter differences, or statistical sampling is used, providing an estimation of the event distribution over the code range. The

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 191–198, 2005.
© Springer-Verlag Berlin Heidelberg 2005

latter is done by interrupting the execution every n-th event, and building up a histogram over the program counter at interrupt times.

The described tools provide code ranges where most of the cache misses are happening, i.e. most of the time is spent for memory-intensive applications. However, cache misses are symptoms of memory accesses with bad locality, and not at the same time necessary their cause: e.g. a non-iterative read-stream may pollute the cache and lead to all the misses in the application, but this may happen because other data may have bad access locality: in this case, the tool points to the wrong position. This example may seem to be contrived, and in fact, usually most memory blocks loaded into cache are used more than once, thus being subject to eviction themselves: i.e. the distribution over block-evicting code positions may be similar to that over block-evicted code positions. Still, by relating cache misses to code positions where the evicted lines were loaded, one level of indirection can be avoided. Fig. 1 shows an optimization which gets rid of two misses, and which needs `a` (or the code accessing `a` to be shown as hint for the programmer. Besides, if a cache is enlarged by a small amount, evictions usually happen at a later point in time, and possibly at other code positions, but evicted lines will be the same.

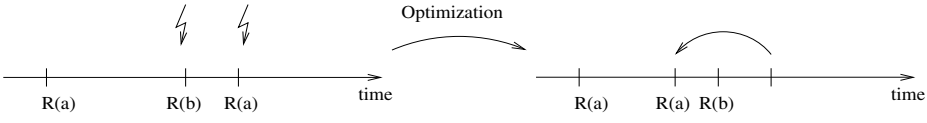


Fig. 1. Optimization by reordering accesses of a (b conflicts with a)

Cache reuse metrics are good to provide more information about cache utilization: the average temporal reuse of a memory block by a cache is given by the number of times the cache was accessed for the memory block divided by the number of times the block was loaded from the next memory hierarchy level. Spatial locality can be measured by the percentage of bytes accessed in a loaded memory block before it is evicted again. Finally, an interesting metric is the stack reuse distance of two sequential accesses to the same memory block: this is the number of memory blocks accessed in-between [3]. This metric, assuming full associativity, is independent of cache size, and gives a precise prediction of when an access will be a cache miss: namely, if the stack reuse distance of this access and the last access to the same location is bigger than the cache size. All these metrics are useful if related to the code positions where the memory block was loaded, and not where the block was evicted (the latter is not applicable to stack reuse distances).

In Chapter 2, we explain the extensions implemented in the cache simulator Callgrind [14], based on the motivation just mentioned. This extension provides metrics for temporal reuse and spatial use of cache lines loaded, related to the code positions where they where loaded. The cache simulator is driven by using runtime instrumentation on the execution of unmodified x86-binaries on Linux

with the help of the Valgrind instrumentation framework [13]. The produced profile data can be visualized without modification in KCachegrind[1]. In Chapter 3, we give some simple examples on the usefulness of the collected metrics for cache optimization. In chapter 4, we present ideas on how to implement the collection of these metrics in hardware, and in Chapter 5, we present some related work. The last chapter gives future research directions.

2 Cache Simulation Additions

Callgrind is an enhanced version of Cachegrind, a cache simulator based on the Valgrind instrumentation framework mentioned above. Only user level code is used to drive the simulation, but this is enough for most applications. In addition, Callgrind builds the call graph of a program on the fly, relates inclusive cost metrics to call chains, and is able to separate cost for different threads. This allows the visualization tool KCachegrind to show exclusive and inclusive metrics for annotated assembler, source, and functions; in addition, it provides an interactive call graph view and a tree-map visualization, showing the inclusive costs graphically in a tree. Extending Callgrind to produce further metrics for source code positions is supported by KCachegrind without any modification.

To allow for cache block utilization metrics and its relation to the code positions that loaded the block, a further data structure was added for every cache block of the L1 and L2. This structure contains the following data for a memory block which currently resides in the corresponding cache:

- the code position of the instruction that recently has filled the block (i.e. the call chain and instruction address),
- a reuse count, and
- a block mask, able to hold the bytes that were touched at least once (a 32bit mask is used, and thus, for 64 byte cache lines, this mask gives a 2-byte granularity).

For every data reference, the runtime instrumentation provides the address and length of the access. These are used in the cache simulation. For an L1 hit, the reuse count and the mask of the according reuse info structure is updated. For an L1 miss, the mask is converted to a byte count, and this and the reuse count are added to counters which are attributed to the stored code position, called "Temporal Reuse L1", and "Spatial Reuse L1". The to-be-evicted block can also be found in the L2. By maintaining according pointers, we can update the reuse metrics for the block in L2: the reuse count is added to the L2 reuse count, and spatial use masks are OR'ed. Afterwards, the L1 reuse counters are initialized for the new data which was loaded on the L1 miss. On an additional L2 miss, the reuse metrics are similarly attributed to counters called "Temporal Reuse L2", and "Spatial Reuse L2".

[1] KCachegrind and Callgrind both are available under the GNU Public License at `http://kcachegrind.sf.net`

It would be quite easy to relate the reuse metrics to code position tuples, adding the code position that evicted the respective cache lines. Currently, this is work in progress on the side of the visualization tool.

3 Results with Example Code

Cache reuse metrics can pinpoint the following problems:

- code regions with low spatial access locality,
- code regions with low temporal locality.

Low spatial and temporal locality is only important if the number of memory accesses in these regions make up a large portion of all accesses of the application run.

By default, the cache simulator uses cache attributes for L1 and L2 which are the same as on the machine where the simulation is run. In the following examples, we use a Pentium M with 32 KB L1 and 1024 KB L2 cache size; line size is 64 bytes and associativity is 8 for both.

The code example for the first problem iterates over a 2D matrix in the wrong index order, thus emphasizing low locality. Table 1 shows the difference of cache reuse metrics annotated to code using the right (1) and wrong (2) index order. Loads for index variables are not visible because of a initialization loop over the matrix, i.e. aside from read accesses to `matrix[i][j]`, no L1 misses are observed. As a lot of columns fit into L2 cache, the L2 load number for the wrong index order is smaller than L2 loads for the correct order, but reuse for L2 is optimal in all cases. So this example shows only bad behavior regarding L1.

Table 1. Iteration over a 2D matrix

	Code	L1 Loads	L1 Spat. Reuse	L1 Temp. Reuse	L2 Loads	L2 Spat. Reuse	L2 Temp. Reuse
	`int matrix[1000][1000]`						
	`...`						
	`for(i=0;i<1000;i++)`	0			0		
	`  for(j=0;j<1000;j++)`	0			0		
(1)	`    sum += matrix[i][j]`	62 500	100 %	16	62 500	100 %	16
(2)	`    sum += matrix[j][i]`	999 992	6 %	1	60 190	100 %	16

The example for low temporal locality does a matrix-matrix multiplication without blocking. Table 2 similarly shows the difference of cache reuse metrics annotated to the unblocked (1) and the blocked (2) version of this code. L2 reuse metrics show a quite good behavior because of the quite small matrix size.

For larger programs, first, one has to locate the code regions with bad locality before being able to look at details like the ones outlined above. One has to find

Table 2. Matrix multiplication unblocked and blocked

Code	L1 Loads	L1 Spat./Temp. Reuse	L2 Loads	L2 Spat./Temp. Reuse
(1) `for(i=0;i<1000;i++)`	0			
`for(j=0;j<1000;j++)`	0			
`a[i][j] += b[i][j] *` `b[j][i];`	1 123 993	68 % /2.65	176 398	94 % /15
(2) `for(ii=0;ii<1000;ii+=50)`	0			
`for(jj=0;jj<1000;jj+=50)`	0			
`for(i=0;i<50;i++)`	0			
`for(j=0;j<50;j++)`	0			
`a[i+ii][j+jj] +=` `b[i+ii][j+jj] *` `b[j+jj][i+ii];`	219 513	81 % / 13.0	175 791	100 % / 16

a derived event type formula using miss counts and reuse metrics, such that sorting the resulting costs over the whole program allows to spot the important bottlenecks easily. To spot code lines with low temporal locality, a ordering of the lines according to the formula

$$MissCount/TemporalReuse$$

should be used, and for low spatial locality

$$MissCount * (1 - SpatialReuse)\,,$$

using the metrics attributed to one line each.

4 Hardware Support

To build caches which are able to monitor cache reuse, the additionally used circuit space should to be minimized. An extension which produces data on every cache miss would almost need a second processor to handle the amount information collected. Thus, counters which are used in a statistical sampling method seem adequate.

We want to relate the reuse metrics to the code position which triggered the load of that cache line. The simplest solution would be to store this code position for every cache line on a cache load. Unfortunately, there is no easy way to get at this address from the cache controller's point of view, in the presence of out-of-order processors. In addition, this would need quite a large additional amount a chip space. But the cache controller could store the old tag on a cache miss, enabling us to obtain the address of the evicted memory block in an interrupt handler which is triggered at overflow time of a hardware performance counter for cache misses. Looking at Intel P4 and Itanium processors, there is

a possibility to get at the data address which leads to the cache miss (PEBS on P4), or to a long cycle stall because of memory access (special register on Itanium, with "EAR" events). This way, our cache reuse metric at least could be attributed to (evicted,evicting)-data address tuples.

However, relation to source code would be desirable. By simultaneously sampling misses related to (code position, evicting address), we get some meaning of which data structures are accessed at which code positions. By presenting this information to the user in a meaningful way, he or she should be able to make use of it.

For the reuse metrics themselves, some bits have to be spent for each cache block: for spatial locality, a granularity at byte level is not necessarily needed: A bit mask with one bit for a floating point value should be sufficient, resulting in 8 bits per 64 byte cache line. For temporal locality, it should be sufficient to use a saturating counter with e.g. 4 bits per cache line, as a reuse of more than 16 probably does not need to be optimized. However, this is arguable and subject to further research. The cache controller has to be improved for updating these masks and counters, and on a miss, a performance counter has to be incremented by the according amount. Presuming that only the additional bits per cache line influence the additional need chip space, 2.3 % of additional space are needed (12 bits per 64*8 bits).

While it is unlikely that a processor vendor would add these bits for reuse metrics on every consumer processor, allowing the reading of the old tag on a cache miss for an interrupt routine seems plausible.

5 Related Work

Most profilers relate timer or performance counter events like cache misses to code positions, like DCPI [1] (for Alpha), VTune [7] (for x86) or OProfile [9] (Linux), most often using statistical profiling. While this can be done without recompiling, instrumentation is needed to get more information. This way, tools can get exact event counts of code regions and call graphs like Adaptor [4] or VTune. Using cache simulation, runtime instrumentation is most comfortable to get cache events, like in Cachegrind [12], which the simulator in this paper is based upon.

Cache simulation currently is the only way to obtain cache utilization metrics. Memspy [10], SIGMA [5] SIP [2] and MHSIM [11] are examples of cache simulators. The latter two are similar to our work; MHSIM is based on source instrumentation, ignoring compiler optimizations. SIP only calculates metrics at one cache level. It runs on SPARC, simulates a whole system, and produces fixed HTML files. Our tool only applies to user level, but its cache simulation can be switched on dynamically only in interesting program phases, and it is as easy to use as `strace` or `perfex`. It takes advantage of the visualization possibilities of KCachegrind.

Regarding hardware support for enhanced performance analysis, Intel's latest processors seem to provide most features. The number of available performance

counters (18) and types of events measurable, in addition to advanced filter and cascading possibilities, together with PEBS (see Chapter 4), make the Pentium 4 platform a good target for any performance analysis (tools to be written) [6]. The Itanium 2 platform at least has a similar level of support for performance analysis tools, as it is able to filter events on opcode, instruction and data ranges, and provides data addresses for data accesses lasting longer than a given threshold. This abstracts from any cache miss events, and focuses on actual latency, which is regarded as valuable.

6 Summary and Future Work

In this paper, we added support for collection of cache reuse metrics to Callgrind, part of our tool suite for cache analysis. The visualization side, KCachegrind, did not need to be changed for this addition.

Work is carried out on being able to visualize metrics related to tuples of code positions. This will allow us to relate reuse metrics to (evicted, evicting) code positions, and enable for addition of the stack reuse distance metric in the tool. Stack reuse distance usually involves $O(\log n)$ complexity for updating a stack of size n for each access, but a constant time algorithm is available for approximated results (buckets of distance ranges), which should be adequate.

In joint work with hardware people in our group, we want to check the usefulness and complications of adding hardware support for reuse metrics as given in Chapter 4, by modifying a VHDL description of a simple MIPS core, and running it in an FPGA.

Acknowledgments to Julian Seward and Nick Nethercote for their useful and "simply working" Valgrind tools.

References

1. J. M. Anderson, L. M. Berc, J. Dean, et al. Continuous Profiling: Where Have All the Cycles Gone? *ACM Transactions on Computer Systems*, 15(4):357–390, November 1997.
2. E. Berg and E. Hagersten. SIP: Performance Tuning through Source Code Interdependence. In *Proceedings of the 8th International Euro-Par Conference (Euro-Par 2002)*, pages 177–186, Paderborn, Germany, August 2002.
3. K. Beyls and E.H. D'Hollander. Platform-Independent Cache Optimization by Pinpointing Low-Locality Reuse. In *Proceedings of International Conference on Computational Science*, volume 3, pages 463–470, June 2004.
4. T. Brandes. Adaptor. Homepage. http://www.scai.fraunhofer.de/291.0.html.
5. L. DeRose, K. Ekanadham, J. K. Hollingsworth, and S. Sbaraglia. SIGMA: A Simulator Infrastructure to Guide Memory Analysis. In *Proceedings of SC 2002*, Baltimore, MD, November 2002.
6. Intel Corporation. IA-32 Intel Architecture: Software Developers Manual.
7. Intel Corporation. Intel VTune Performance Analyser. available at http://www.intel.com/software/products/vtune/.

8. M. Kowarschik and C. Weiß. An Overview of Cache Optimization Techniques and Cache-Aware Numerical Algorithms. In U. Meyer, P. Sanders, and J. Sibeyn, editors, *Algorithms for Memory Hierarchies — Advanced Lectures*, volume 2625 of *Lecture Notes in Computer Science*, pages 213–232. Springer, March 2003.
9. J. Levon. OProfile, a system-wide profiler for Linux systems. Homepage: http://oprofile.sourceforge.net.
10. M. Martonosi, A. Gupta, and T. E. Anderson. Memspy: Analyzing memory system bottlenecks in programs. In *Measurement and Modeling of Computer Systems*, pages 1–12, 1992.
11. J. Mellor-Crummey, R. Fowler, and D. Whalley. Tools for Application-Oriented Performance Tuning. In *Proceedings of 15th ACM International Conference on Supercomputing*, Italy, June 2001.
12. N. Nethercote and A. Mycroft. The Cache Behaviour of Large Lazy Functional Programs on Stock Hardware. In *Proceedings of the ACM SIGPLAN Workshop on Memory System Performance (MSP 2002)*, Berlin, Germany, July 2002.
13. N. Nethercote and J. Seward. Valgrind: A Program Supervision Framework. In *Proceedings of the Third Workshop on Runtime Verification (RV'03)*, Boulder, Colorado, USA, July 2003. Available at http://developer.kde.org/~sewardj.
14. J. Weidendorfer, M. Kowarschik, and C. Trinitis. A Tool Suite for Simulation Based Analysis of Memory Access Behavior. In *Proceedings of International Conference on Computational Science*, volume 3, pages 455–462, June 2004.
15. Wm. A. Wulf and S. A. McKee. Hitting the Memory Wall: Implications of the Obvious. *Computer Architecture News*, 23(1):20–24, March 1995.

Modelling and Animating Hand Wrinkles

X. S. Yang and Jian J. Zhang*

National Centre for Computer Animation,
Bournemouth University, United Kingdom
{xyang, jzhang}@bournemouth.ac.uk

Abstract. Wrinkles are important visual features on the skin of an animated character. However, realistic animation of wrinkles is not easy, especially for the hand, as it gathers many types of different wrinkles. In this paper, we present a computationally efficient method for the generation of hand wrinkles, both for the palm and the fingers. This method is developed by exploiting the inherent properties relating to wrinkles, which are volume preservation of the underlying flesh and incompressibility of the skin. Although this method is for the hand model, it is also applicable to other parts of the body where the skin is deformed by underlying skeleton joints.

1 Introduction

Wrinkles are important visual features for realistic animated characters. On a human body, there are two types of wrinkles: fine-scale and large-scale ones. Between them, the latter is more significant in computer animation, as the large-scale wrinkles mark pronounced feature lines on the skin surface and are obvious to the human eye. The large-scale wrinkles change with the motion of the character and capture the motion related skin deformations. The fine-scale ones are, on the other hand, much less visible and are relatively static compared with the other type.

There have been a number of techniques developed in representing and animating wrinkles on an animated human character. The fine-scale wrinkles are usually produced as texture on the skin mesh. As long as the skin area concerned does not deform to a large amount, this technique is effective and is reasonably realistic. Physiologically, large-scale wrinkles on a human body were there when the person was born, starting with shallow lines. These lines get deeper with the repetitive motion of the body parts, such as those on the forehead and the palm. Human skin deforms due to the properties of the skin and the underlying internal body elements. Distinctive skin deformation patterns are formed around wrinkles as a result of two most significant features relating to the human skin. The first is the incompressibility of the skin and the other is the volume preservation property of the underlying flesh. To animate a human character

* Corresponding author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 199–206, 2005.
© Springer-Verlag Berlin Heidelberg 2005

realistically, these properties need to be convincingly modelled. This accounts for the difficulty for simulating large-scale wrinkles realistically in computer animation.

In this paper, we concentrate on the problem of large-scale wrinkle modelling, in particular, we will look into how hand wrinkles can be modelled and animated realistically within current animation production pipeline. Because of the properties of the skin around a wrinkle, physically simulating the deformation process is no doubt ideal. Nevertheless, such an approach often involves a sophisticated mechanical model and is likely to involve heavy computational overheads. Since what matters to animation is visual realism and in many case computational efficiency, in this paper we will develop a non-physically based simulation technique. The idea is to reproduce the skin deformation parametrically around large-scale wrinkles.

The remainder of the paper is structured as follows. Previous work was briefly reviewed in Section 2. Section 3 discusses the issues relating to a wrinkle deformation region including mesh generation. In Section 4.1, the shape of the cross section of large-scale wrinkles is derived. This is then followed by wrinkle animation in Section 4.2. Section 5 concludes the paper.

2 Previous Work

The most straight forward idea to generate wrinkles is texture mapping. The texture can be either extracted directly from photos or synthesized. However, it is difficult to animate, especially when large deformation is involved. Viaud and Yahia [1] presented one of the early methods for wrinkle modelling. They modelled wrinkle bulges as spline segments. Bando et al [2] presented a technique for the modelling of both fine-scale and large-scale wrinkles. For the fine-scale ones, their method is in essence a texture synthesis technique. For large-scale wrinkles, they displace the vertices concerned in the normal direction of the wrinkle shape. Adaptive mesh refinement is carried out for those areas if the mesh is too coarse for the required resolution. Larboulette and Cani [3] recently presented a technique for generating dynamic wrinkles geometrically. Their idea is to model wrinkles as a length preserving curve. When a curve is pushed at one end, wrinkles start to appear and ripple through the whole length. Because of this analogy, their method is effective on parallel wrinkles appearing on both cloth and human skin.

Physics-based techniques have also been studied by many researchers [4]. Wu's method [5] includes both texture mapping and a linear plastic model to simulate the wrinkling phenomenon. Boissieux et al. [6] modelled skin as a volumetric substance with a layered structure. The finite element method was used to compute the deformation. In general, physical modelling methods tends to produce mathematically more accurate results, however, are in practice more expensive than other alternatives.

3 Wrinkle Deformation Regions

On the palm and other parts of the human body where large-scale wrinkles appear, the skin deforms locally when the relevant body parts move. In this Section, we present a method to define a deformation region for each or a group of wrinkle(s). We also discuss how this region is seamlessly merged with the existing surface mesh of the animated character model.

3.1 Wrinkle Lines

Wrinkle lines define the direction and pattern of the wrinkles. These lines are user-specified and can be either manually drawn or obtained from other techniques. Essentially, they reflect the intention of the user (animator). Once these lines are given, our technique will then produce the mesh geometry and deformation following the animation parameters. Figure 1 shows some typical wrinkle lines on the palm, each line consists of a number of segments.

Fig. 1. Hand-drawn wrinkle lines

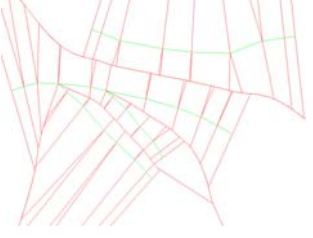


Fig. 2. Voronoi diagram and wrinkle lines

Fig. 3. Wrinkle lines and deformation region

3.2 Determination of Wrinkle Deformation Regions

Media axes are feature lines which can partition an irregular region with some sense of 'symmetry' [7]. From observing skin deformation resulting from wrinkles, we believe it is reasonable that their deformable region can be defined as an area where the wrinkles are regarded as their media axes.

Practically, we try to define an offset curve for each wrinkle line on the parametric space of the existing model. As each wrinkle line is a separate line entity, its offset curve actually marks out a closed region. The overall deformation region for a group of connected and closely laid wrinkle lines are taken as the union of all the separate regions.

To compute the boundary and mesh of this region, we construct a Voronoi diagram. Algorithmically, for each wrinkle line, we have adopted the algorithm from [7]. Figure 2 shows the Voronoi diagram (red) corresponding to the wrinkle lines (green). By considering the influence distance, we are able to mark the deformation region out of the diagram, as shown in Figure 3.

3.3 Mesh Generation

One of the advantages of using a wrinkle deformation region is that we can avoid mesh refinement, which can be expensive, as the original mesh usually does not guarantee that sufficient number of mesh vertices are placed within the wrinkle area. Mesh subdivision is necessary to produce a large number of densely populated vertices. This however, can be unnecessarily expensive. Using the deformation regions presented in above, the mesh is ready in place to deform the skin in a required manner.

By superimposing Figure 2 with Figure 3, we can generate the mesh in the deformation region, as seen in Figure 4. The vertices are in the right place to represent wrinkle deformation. From our experiment, this initial mesh is adequate for most wrinkle animation applications.

Now we are ready to merge the deformation region with the animation model, in our case, the hand model. This is a straightforward process. What we need to do is to place the deformation region on the parametric space of the model and remove all vertices falling into the region from the model. The deformation region carved out of the model mesh is patched up by the Voronoi mesh (Figure 5a). At the boundary of the deformation region, vertices from the original model mesh and the Voronoi mesh may not match exactly. Triangulation around the boundary is necessary to fix it. Figure 5b shows the mesh of the wrinkle deformation region merged with the original hand mesh. Notice that the mesh does not need to be dense.

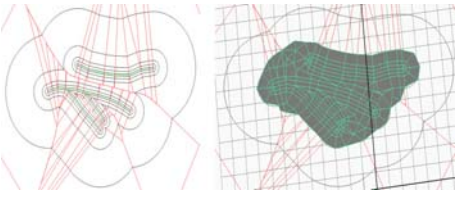
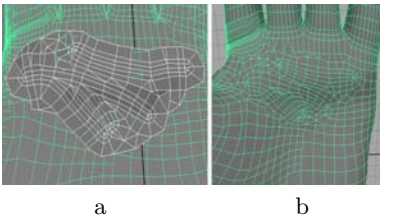


Fig. 4. Initial mesh in the deformation region



a b

Fig. 5. Merge both meshes together

4 Animation of Wrinkles

The animation of wrinkles involves two key elements, deformation simulation conforming to the skin properties around wrinkles; and the establishment of the relationship between deformation and motion parameters.

4.1 Skin Deformation

Realistic skin deformation relies on the simulation process to recognise the properties of the skin around a wrinkle. As the visual quality is what we are after rather than the physical accuracy, we do not intend to simulate the phenomenon

based on the physical principles that govern the deformation. Instead, we will use a cheaper alternative to formulate the shape of the skin.

Two properties of the human skin are of particular importance to achieve good visual quality of an animated character, which are volume preservation of the tissue under the skin; and the incompressibility of the skin itself. In the following we will develop equations that are able to capture the essence of these properties.

Volume Preservation. In the cross section of a wrinkle, the profile is similar to the curve given in Figure 6. At its rest pose the initial values of both parameters, h and a, are given by the user. Assuming the skin was flat before a wrinkle was placed, the presence of the wrinkle ditches the skin, but the flesh is bulged at the sides.

We propose a piecewise polynomial function, called the shape function, to represent the two sections illustrated by the profile curve in Figure 6. The two sections are $[0, 2a]$ and $[2a, A]$. Both connect smoothly at $x = 2a$ when the shape function peaks. From Figure 6, the constraints for the first section are that the curve passes three points $(0, -h), (a, 0)$ and $(2a, h/8)$, and satisfies $f'(2a) = 0$. Thus the shape function for the first section is given by:

$$f(x) = \frac{5h}{32a^3}x^3 - \frac{29h}{32a^2}x^2 + \frac{7h}{4a}x - h \;\; x \in [0, 2a] \tag{1}$$

For the second section $[2a, A]$, the shape function is developed from the cubic Hermite interpolation method, where unknown A is to be determined:

$$f(x) = \frac{h}{4(A-2a)^3}x^3 - \frac{3h(A+2a)}{8(A-2a)^3}x^2 + \frac{3hAa}{2(A-2a)^3}x - \frac{-hA^3 + 6hA^2a}{8(A-2a)^3} \;\; x \in [0, 2a] \tag{2}$$

In order to preserve the volume during deformation, the following condition is to be met:

$$\int_0^A f(x)dx = 0 \tag{3}$$

This gives:

$$\int_0^A f(x)dx = \int_0^{2a} f(x)dx + \int_{2a}^A f(x)dx = -\frac{7ha}{24} + \frac{h(A-2a)}{16} = \frac{h(-20a+3A)}{48} = 0 \tag{4}$$

Solving this integral equation, we obtain:

$$A = \frac{20}{3}a \tag{5}$$

Substituting Eq. (5) back into Eq. (2), the function of the second section arrives. In fact, parameter A can also be controlled by the user if this is the user chooses to do. In that case, the volume may vary to suit the intended design of the skin shape.

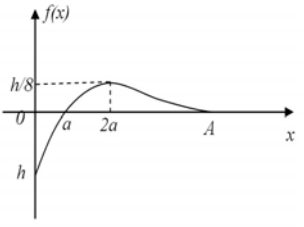


Fig. 6. Wrinkle shape function

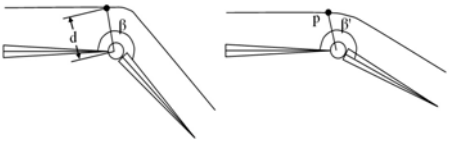


Fig. 7. Skeleton driven-animation

Incompressibility of Skin. Incompressibility is a different property to that of volume preservation. This property suggests that the length of the skin remains constant during animation. Compared with the former, this property is of more significance in visual realism. Linking with the shape function, this condition can be expressed by the following integral equation:

$$G(h,a) = \int_0^A \sqrt{1+[f'(x)]^2}dx = C \tag{6}$$

where C is a constant, which is calculated at the rest pose of the animated model. Since this equation involves two parameters a and h, once the length C is known, both parameters become mutually dependent of each other. Animating the model deforms the wrinkle profile, this further leads to a new value for parameter a (how to compute the value of a is explained at the next subsection). Thus Eq. (6) reduces to $G(h) = C$. This equation is not usually solvable analytically, but can be solved reasonably straightforwardly using a numerical method, such as the Newton method. This produces a new depth parameter h for the new pose. As this is a one-dimensional problem, the resolution is fast. Real-time performance is achieved for all our experiments.

4.2 Wrinkle Animation

Skeleton-driven animation is the most popular technique for character animation, which we use here as the driving action for wrinkle animation. That said, when the skeleton moves, the character model, such as a hand model, together with the wrinkles on the skin surface is animated. What we are trying to work out is how such skeleton-driven animation is related to the shape change of the skin, and in turn to the change of the wrinkling parameters, a and h, in our case.

Most hand joints have one dominating rotational degree of freedom (DOF), such as the fingers and palm. In fact, even for joints with a complex combination of angular movements, at an instant, it is reasonable to regard the rotational motion as being of one DOF, perhaps in an arbitrary orientation. This observation is useful in simplifying the animation process.

Let us look at the wrinkles around a finger joint, where the joint angle acts as the driving parameter. If it bends, the back of the finger stretches and the

inside surface shrinks, which makes the wrinkle around the joint more or less prominent. Figure 7 shows the skeleton and skin surface around a joint. P is a point on the wrinkle line, d is the distance from the point to the joint centre, β is the joint angle at the rest pose. When the joint rotates to angle β', the length of the skin surface is shortened and the shrinkage can be computed by:

$$S = d * \frac{\beta'}{\beta} \tag{7}$$

Here S indicates the amount of shortening of the skin relative to its original length. $S = 1$ means it is the rest pose, $S > 1$ suggests the skin expands, and $S < 1$ signals the skin shrinks. Thus a corresponding new value of a is given by:

$$a = S * a_0 \tag{8}$$

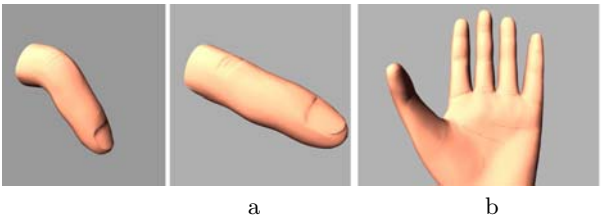
where a_0 is the parameter at its rest pose.

With a we can compute a new value of h using Eq. (6), if accuracy is important. Otherwise we can simply approximate the shape function curve between $[0, a]$ by a straight line whose length is kept constant. This results in a simpler formula for the depth h:

$$h = \begin{cases} \sqrt{h_0^2 + (1 - s^2)a_0^2} & s < \sqrt{1 + \frac{h_0^2}{a_0^2}} \\ 0 & s \geq \sqrt{1 + \frac{h_0^2}{a_0^2}} \end{cases} \tag{9}$$

where h_0 denotes the parameter at the rest pose. This is reasonable, as when the skin exceeds the length of the skin when being stretched, the depth of the wrinkle will become zero. When the skin shrinks, the depth will increase reaching the maximum depth of $\sqrt{h_0^2 + a_0^2}$.

Figure 8(a) shows the animation of a finger and the wrinkles on the back surface. Figure 8(b) demonstrates the wrinkles generated on the palm and at the joints of the fingers.



a b

Fig. 8. Application of wrinkles: (a) wrinkled finger model (b)Wrinkles on a hand model

5 Conclusion

Wrinkles are important visual features on the skin of an animated human and creature character. In this paper, we have presented a new method for the gener-

ation of hand wrinkles quickly and realistically. This method works in the following steps. Wrinkle lines are given at the first step, which are to catch the intent of the animator. Such wrinkle lines can be either manually drawn or scanned in. In the second step, a deformation region is defined for a group of closely laid wrinkle lines. Local meshes within the deformation regions are produced and merged with the mesh of the model. Compared with other existing techniques, our method is cheaper and needs a smaller number of vertices in the deformation region. A wrinkle shape function is derived in the third step, where both volume preservation and skin incompressibility are taken into account. Due to the exploitation of these two properties, our method is able to produce visually realistic results without using full scale physics. In the final step, the wrinkles are animated together with the hand model, where key parameters of the wrinkle shape function are determined at each pose.

Although our method uses the hand model explicitly, it is applicable also to other parts of the body where the skin is deformed by underlying skeleton joints.

Acknowledgements

This research is funded by the AHRB grant B/RG/AN5263/APN12727. The authors would also like to gratefully acknowledge Martin Held for providing us the code to generate Voronoi diagrams. We are grateful for the donation of the Maya licenses from Alias.

References

1. M.L. Viaud, H.Y.: Facial Animation with Wrinkles. Proc. Eurographics workshop on Animation and Simulation. (1992).
2. Bando, Y.T.U., Japan; Kuratate, T.; Nishita, T.: A simple method for modeling wrinkles on human skin. Proceedings 10th Pacific Conference on Computer Graphics and Applications. (2002)
3. Larboulette, C. and M.-P. Cani: Real-Time Dynamic Wrinkles, Computer Graphics International. (2004) IEEE Computer Society Press.
4. Wu, Y., N.M. Thalmann, and D. Thalmann: A plastic-visco-elastic model for wrinkles in facial animation and skin aging, Proceeding of the second Pacific conference on Fundamentals of computer graphics. (1994) 201–213.
5. Yin Wu, P.K., Laurent Moccozet , Nadia Magnenat-Thalmann, Simulating wrinkles and skin aging. The Visual Computer, **15**(4)(1999) 183 - 198.
6. Boissieux, L.K.; Thalmann, N.M.; Kalra, P.: Simulation of skin aging and wrinkles with cosmetics insight. in Computer Animation and Simulation, Eurographics, (2000).
7. Held, M., VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. Computational Geometry, **18**(2)(2001) 95-123.

Simulating Wrinkles in Facial Expressions on an Anatomy-Based Face

Yu Zhang, Terence Sim, and Chew Lim Tan

School of Computing, National University of Singapore, Singapore 117543

Abstract. This paper presents a muscle-driven wrinkle model for simulating dynamic wrinkles that appear during facial expressions, which is geometric in nature. Wrinkles are generated on an anatomy-based face model that incorporates a structure of skin, muscles and skull for physically-based animation. Corresponding to two types of facial muscles, a geometric model is developed to govern how the wrinkle amplitude evolves locally upon skin deformation, taking into account the properties of real wrinkles. During facial animation, wrinkles are generated in the local regions influenced by muscle contraction, simulating resistance to compression of tissues. Using our method, realistic wrinkles can be dynamically rendered on the adaptively refined face mesh at interactive rates.

1 Introduction

Facial expressions exhibit not only facial feature motions, but also subtle changes in skin appearance such as facial creases and wrinkles. These details, or expressive wrinkles, constitute major elements affecting the visual appearance of the face. Therefore, realistic modeling of expressive wrinkles contributes significantly to the overall realism of facial animation.

One class of methods to generate facial wrinkles is the texture mapping approaches such as bump texture mapping and color texture mapping [2, 5, 6]. Although being efficient and giving acceptable visual results in simulating facial wrinkles formed due to skin texture and age, this technique has some inherent limitations. First, it is difficult to mimic the wrinkle dynamics during animation. Second, as the skin geometry remains undeformed, the perceptible geometrical and structural changes can not be modeled.

Another class of techniques, known as physically-based methods [3, 7, 11, 12], attempt to model the dynamic wrinkling as the result of skin deformation by approximating the biomechanical properties of real skin. Results of the physically-based methods are realistic, but the process is not real-time due to expensive computation of the underlying mass-spring or finite element system.

Geometric methods solve the difficulties associated with two classes of methods mentioned above by simulating the perceptible geometrical deformation of skin for rendering at a high simulation speed [1, 4, 8, 9]. Although these methods are efficient to produce convincing results, they reveal the difficulties in intuitive control such as adjusting size and position of wrinkles and involve painstaking tasks such as manually drawing wrinkle patterns in textures. Since expressive wrinkles are formed due to compression of the skin surface as a result of muscle contraction, our goal is to easily model wrinkles according to the influence of virtual facial muscles.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 207–215, 2005.
© Springer-Verlag Berlin Heidelberg 2005

In this paper, we present a muscle-driven wrinkle model from an anatomic point of view for efficiently simulating expressive wrinkles. Our method generates wrinkles on an anatomy-based face model. Corresponding to two types of facial muscles, a geometric model is developed to govern how the wrinkle amplitude evolves locally upon skin deformation, taking into account the properties of real wrinkles. During facial animation, wrinkles are generated in the local regions influenced by muscle contraction, simulating resistance to compression of tissues. Our method simulates wrinkles that can be dynamically rendered on the adaptively refined face mesh at interactive rates.

2 Muscle-Driven Modeling of Wrinkles

2.1 An Anatomy-Based Face Model

We have developed an anatomy-based face model for physically-based facial animation [13]. The model encapsulates three structural layers: skin, muscles, and skull (see Fig. 1). The reader is referred to [13] for technical details.

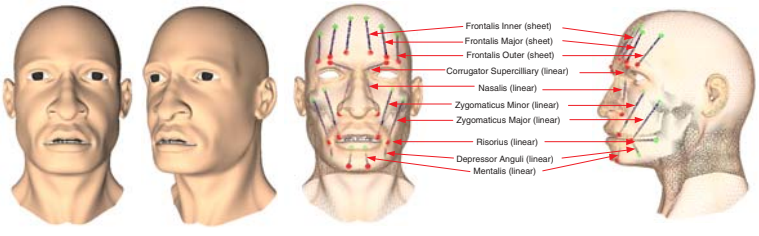


Fig. 1. The anatomy-based face model. Left: face geometry. Right: multi-layer anatomic structure of the skin, muscles and skull

Since expressive wrinkles are dynamically generated due to shrinking of the skin surface as a result of muscle contraction, we propose a *muscle-driven wrinkle model* to automatically simulate them. In our face model, we use two kinds of physically-based facial muscle models that are based on Waters' muscle model [10]: the *linear muscle* for the majority of facial muscles, the *sheet muscle* for the forehead muscles (Frontalis).

2.2 Modeling of Wrinkles Driven by the Linear Muscle

Muscle Model and Its Wrinkling Region. A linear muscle is defined by an insertion point, the end of the muscle attached to skin; an attachment point, the end of the muscle fixed to bone; and the area of influence (AOI), the portion of the surface affected by the muscle contraction. The effect of facial muscle contraction is to pull the facial skin from the area of the muscle insertion point to the static muscle attachment point on the bone. Fig. 2 illustrates the linear muscle model and its AOI with the following definitions:

$\mathbf{x}_i$: arbitrary facial skin point
$\mathbf{f}_i$: muscle force applied to $\mathbf{x}_i$
$\mathbf{m}_A$: attachment point of linear muscle at the skull
$\mathbf{m}_I$: insertion point of linear muscle at the facial skin
ϕ : maximal angle of muscle influence
φ_i : angle between the central muscle fiber $\mathbf{m}_A\mathbf{m}_I$ and $\mathbf{m}_A\mathbf{x}_i$
R : maximal radius of influence
l_i : distance between $\mathbf{x}_i$ and $\mathbf{m}_A$ projected onto the central muscle fiber $\mathbf{m}_A\mathbf{m}_I$

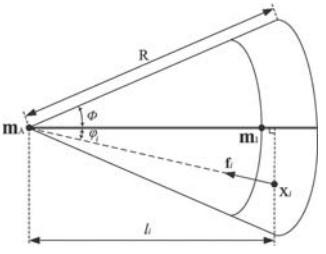


Fig. 2. Linear muscle model

For an arbitrary vertex $\mathbf{x}_i$ on the skin mesh, its distance from the muscle attachment point projected onto the central muscle fiber $\mathbf{m}_A\mathbf{m}_I$ is calculated as:

$$l_i = \|\mathbf{m}_A - \mathbf{x}_i\| \cos(\varphi_i) \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm. l_i and φ_i weight the influence of the linear muscle at vertex $\mathbf{x}_i$ separately for length factor λ_i and angular factor γ_i:

$$\lambda_i = \frac{l_i}{R}, \qquad \gamma_i = \frac{\varphi_i}{\phi} \tag{2}$$

λ_i defines the normalized longitudinal distance of vertex $\mathbf{x}_i$ to the muscle attachment point. γ_i is defined in [0,1] and represents the distance between $\mathbf{x}_i$ and the central muscle fiber in the angular direction. The wrinkling region of the influenced skin surface is a sub-area of muscle's AOI. The set of vertices inside the wrinkling region is

$$S_w = \{\mathbf{x}_i | \lambda_w \leq \lambda_i \leq 1 \quad \text{and} \quad 0 \leq \gamma_i \leq 1\} \tag{3}$$

where λ_w is a user-specified factor defined in [0,1] to control the size of the wrinkling region. The smaller λ_w, the larger the wrinkling region.

Geometric Wrinkle Model. The wrinkle amplitude is calculated for all vertices that are originally (before activating the muscle contraction) inside the wrinkling region. The wrinkle amplitude at a vertex $\mathbf{x}_i$ is given as *wrinkle amplitude function* $W(\mathbf{x}_i)$:

$$W(\mathbf{x}_i) = H(w_i)A(\varepsilon_i)At(\varphi_i) \tag{4}$$

where $H(w_i)$ is the *shape function* used to obtain natural wrinkle shape, *anisotropic function* $A(\varepsilon_i)$ modulates the wrinkle amplitude to model irregularity of wrinkles, and $At(\varphi_i)$ is the *attenuation function*.

Wrinkle Shape

The cross sectional shape of the wrinkle is defined by the shape function $H(w_i)$, a height function of 3D position of the vertex:

$$H(w_i) = \begin{cases} k_0(1+(\frac{w_i-u}{u})^2)^\alpha(\frac{w_i-u}{u})+d & 0 \le w_i \le u \\ -k_0(1+(\frac{w_i-u}{u})^2)^\alpha(\frac{w_i-u}{u})+d & u < w_i \le 2u \end{cases} \tag{5}$$

The $H(w_i)$ curve depicted in Fig. 3 illustrates how wrinkling amplitude evolves. In Eq. 5, u is the half cycling length. Given a predefined value to the number of wrinkles (n_w) generated by the contraction of the muscle, u is obtained as:

$$u = \frac{(1-\lambda_w)R}{2n_w} \tag{6}$$

In Eq. 5

$$w_i = l_i - \lambda_w R - \text{int}(\frac{l_i - \lambda_w R}{2u})2u \tag{7}$$

As int(·) truncates a real number to the biggest integer number that is smaller than it, Eq. 7 periodically maps l_i into [0,2u] with a frequency of n_w.

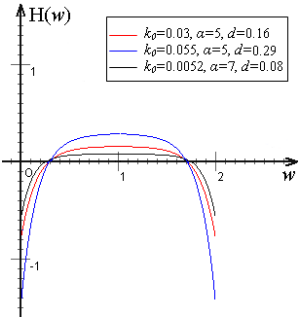


Fig. 3. The shape function with different parameter values (u=1.0)

Three intuitive parameters are introduced in the shape function to control wrinkle characteristics. The parameter k_0 controls the depth of the furrow in a wrinkle and hence the name *furrow depth factor*, α is the *shape modulation factor* that controls the modulation of the wrinkle shape, and d is the *bulge height parameter* that defines the maximal height of the bulge in a wrinkle. By adjusting them, shape function $H(w_i)$ can be chosen to simulate the real wrinkle property: the furrow of a wrinkle is sharp and the bulge is round. Moreover, the shape function should satisfy the following equation:

$$\int_0^{2u} H(w)dw = 0 \tag{8}$$

This means that the volume depressed at the furrow goes to the side to make up the bugle, which simulates the incompressibility of skin. Using Eq. 8, we can express the bulge height parameter d as a function of k_0 and α, which reduces the parameter tuning.

Modulation of Wrinkle Amplitude

Obviously, expressive wrinkles do not all react in the same way to skin deformation, varying according to their alignment with the direction of skin compression. An anisotropic function $A(\varepsilon_i)$ is introduced, which modulates wrinkle amplitude according to skin surface deformation. Suppose a vertex $\mathbf{x}_i$ is connected to one of its neighbors $\mathbf{x}_j$ by an edge with the rest length L_{ij} (see Fig. 4). During animation, we compute at each time step the edge shrinkage τ_{ij} for the edge $\mathbf{x}_i\mathbf{x}_j$:

$$\tau_{ij} = \frac{L_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|}{L_{ij}} \tag{9}$$

τ_{ij} is the amount of shortening relative to the original length, which equals 0 in the rest state. The skin is deformed in the direction of muscle contraction and wrinkles become more pronounced when the skin surface shrinks perpendicularly to their directions. Therefore, the more the direction of an edge runs parallel to the direction of muscle contraction, the more the wrinkle amplitude will react to variation in that edge's length. The compression rate ε_i at the vertex $\mathbf{x}_i$ is defined as:

$$\varepsilon_i = \frac{\sum_{j\in\Omega_i} \tau_{ij}\cos(\theta_{ij})}{\sum_{j\in\Omega_i} \cos(\theta_{ij})} \tag{10}$$

where Ω_i denotes the set regrouping all neighboring vertices that are linked to $\mathbf{x}_i$, θ_{ij} is the angle between edge $\mathbf{x}_i\mathbf{x}_j$ and the direction of muscle contraction $\mathbf{x}_i\mathbf{m}_A$ (see Fig. 4). The weight $\cos(\theta_{ij})$ measures the direction similarity of the considered edge to the muscle contraction direction.

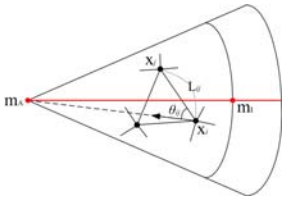


Fig. 4. Shorting of an edge contributes to the compression of the skin surface at a vertex

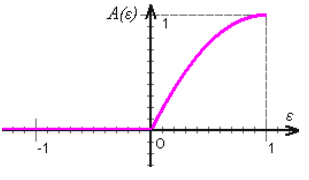


Fig. 5. Plot of the anisotropic function

The wrinkle amplitude varies from maximal amplitude for maximal compression ($\varepsilon_i = 1$) to null for expansion. Negative value of ε_i means that stretching tends to make the wrinkles disappear. We model it by defining the following anisotropic function:

$$A(\varepsilon_i) = \begin{cases} \cos(\frac{\pi}{2}(1-\varepsilon_i)) & 0 \le \varepsilon_i \le 1 \\ 0 & \varepsilon_i < 0 \end{cases} \tag{11}$$

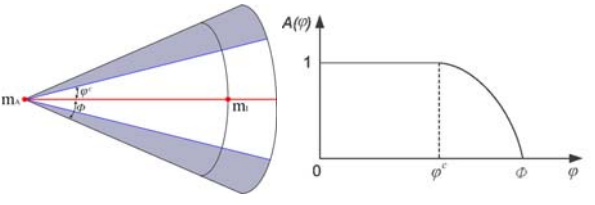
The function curve is depicted in Fig. 5.

Attenuation of Wrinkle Amplitude

The wrinkle amplitude is also scaled by the attenuation function $At(\varphi_i)$:

$$At(\varphi_i) = \begin{cases} 1 & 0 \le \varphi_i \le \varphi^c \\ -\frac{1}{2}(1 + (\frac{\varphi_i - \varphi^c}{\phi - \varphi^c})^2)(\frac{\varphi_i - \varphi^c}{\phi - \varphi^c}) + 1 & \varphi^c < \varphi_i \le \phi \end{cases} \tag{12}$$

where φ_i is the angular distance of a vertex from the central muscle fiber $\mathbf{m}_A\mathbf{m}_I$. $At(\varphi_i)$ defines an attenuation profile which dictates the way the wrinkle height decreases and then vanishes when wrinkles go away from the central muscle fiber. Parameter φ^c controls the angular distance from the central muscle fiber where the wrinkle amplitude begins to decrease (see Fig. 6 (a)). Fig. 6 (b) shows the graph of $At(\varphi_i)$.



(a) (b)

Fig. 6. (a) Attenuation area (gray area) in the wrinkling region of the linear muscle. (b) The attenuation function

2.3 Modeling of Wrinkles Driven by the Sheet Muscle

Muscle Model and Its Wrinkling Region. A sheet muscle is a series of almost-parallel fibers spread over a rectangular area and it contracts to a group of separated muscle fiber nodes. The obvious example is the Frontalis, which lies on the forehead. Fig. 7 illustrates the AOI of the sheet muscle model.

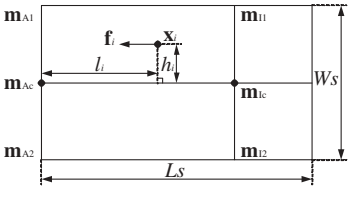


Fig. 7. Sheet muscle model

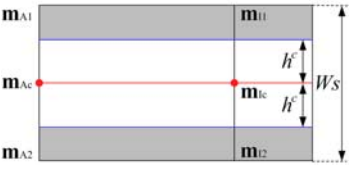


Fig. 8. Attenuation area (gray area) in the wrinkling region of the sheet muscle

Two points $\mathbf{m}_{A1}$ and $\mathbf{m}_{A2}$ specify the attachment line of the sheet muscle. $\mathbf{m}_{Ac}$ is the middle point of the attachment line. Similarly, the points $\mathbf{m}_{I1}$ and $\mathbf{m}_{I2}$ specify the insertion line of the sheet muscle, and $\mathbf{m}_{Ic}$ is the middle point of the insertion line. Vector $\mathbf{m}_{Ac}\mathbf{m}_{Ic}$ is the central muscle fiber. The denotations of other parameters are

$\mathbf{x}_i$: arbitrary facial skin point
$\mathbf{f}_i$: muscle force applied to $\mathbf{x}_i$

Ls and Ws : length and width of sheet muscle's AOI, respectively
l_i : distance between $\mathbf{x}_i$ and the muscle attachment line
h_i : distance between $\mathbf{x}_i$ and the central muscle fiber

Similar to the linear muscle model, the length factor λ_i and width factor γ_i associated with a vertex $\mathbf{x}_i$ are defined as:

$$\lambda_i = \frac{l_i}{Ls}, \qquad \gamma_i = \frac{2h_i}{Ws} \tag{13}$$

The wrinkling region is a sub-area of sheet muscle's AOI. The set of vertices inside the wrinkling region is

$$S_w = \{\mathbf{x}_i | \lambda_w \leq \lambda_i \leq 1 \quad \text{and} \quad 0 \leq \gamma_i \leq 1\} \tag{14}$$

where λ_w in the range [0,1] controls the size of the wrinkling region.

Geometric Wrinkle Model. The wrinkle amplitude function defined in Eq. 4 is employed for wrinkle generation, after modification of a few parameters. For the shape function $H(w_i)$, parameter R in Eq. 6 and 7 is substituted by Ls, the length of sheet muscle's AOI. For the anisotropic function $A(\varepsilon_i)$, since the direction of muscle contraction is parallel to the central muscle fiber $\mathbf{m}_{Ac}\mathbf{m}_{Ic}$, θ_{ij} in Eq. 10 becomes the angle between the mesh edge and $\mathbf{m}_{Ac}\mathbf{m}_{Ic}$. The attenuation function is modified as:

$$At(h_i) = \begin{cases} 1 & 0 \leq h_i \leq h^c \\ -\frac{1}{2}(1 + (\frac{h_i - h^c}{Ws/2 - h^c})^2)(\frac{h_i - h^c}{Ws/2 - h^c}) + 1 & h^c < h_i \leq Ws/2 \end{cases} \tag{15}$$

where h_i is the distance between the vertex and the central muscle fiber, h^c controls the starting position of attenuation (see Fig. 8).

3 Experimental Results

Fig. 9 shows examples of some facial expressions simulated with and without wrinkles, and compares them with the actual ones. Each expression is generated by activating a group of muscles:

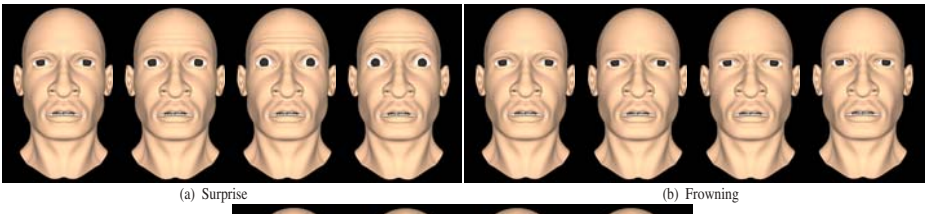
- **Surprise:** Contractions of pairs of Frontalis Major, Frontalis Inner, and Frontalis Outer (sheet muscles) raise the eyebrows and cause wrinkles on the forehead.
- **Frowning:** Contractions of the pair of Corrugator Supercilliary (linear muscle) result in frowning of the eyebrows and large-scale wrinkles between the eyebrows.
- **Fear:** Frontalis Inner and Corrugator Supercilliary are activated to raise the inner portion of eyebrows and to stretch eyebrows towards the face center, respectively.

Fig. 10 shows the snapshots of dynamic wrinkle generation within different expressions. Results reveal that the generated expressive wrinkles satisfactorily enhance the realism of facial animation, making the simulated expressions more convincing.

(a) Surprise (b) Frowning

(c) Fear

Fig. 9. Facial expressions simulated on the face model. Left to right: expression without wrinkles, expression with wrinkles, actual expression generated by a subject (each example)



(a) Surprise (b) Frowning

(c) Fear

Fig. 10. Dynamic simulation of the expressive wrinkles

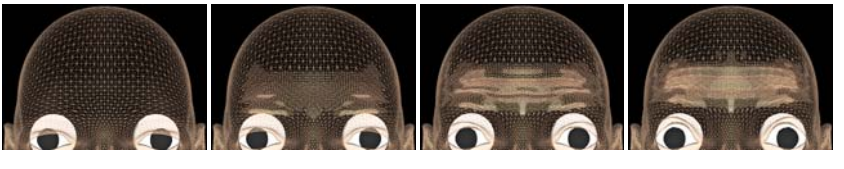


Fig. 11. Adaptive refinement of the face mesh in dynamic wrinkle simulation

The original face mesh consists of about 4,500 triangles. In order to correctly capture the wrinkle shapes, we implement our wrinkling process using a dynamically adaptive refinement scheme of the original mesh. The adaptive refinement is based on a recursive

quaternary subdivision that is applied on the fly where surface curvature (measured from the vertex surface normals) of influenced triangles is too large. Fig. 11 shows a mesh detail of the deformed forehead region during simulation of surprise.

4 Conclusion

This paper has presented a geometric muscle-driven model for simulation of expressive wrinkles. Expressive wrinkles are automatically defined according to facial muscle anatomy. A geometric model governs evolution of wrinkle amplitude in the local deformed regions, taking into account the properties of real wrinkles. It provides intuitive parameters for easy control over wrinkle characteristics and is adapted to two types of facial muscles. Thanks to its geometric nature, our method simulates wrinkles that can be dynamically rendered at interactive rates.

For future work we will introduce a collision detection procedure in our implementation to prevent neighboring folds from intersecting. We would also like to develop the wrinkle model corresponding to the sphincter muscles that surround the eyes and mouth for simulating more realistic and complex deformations.

References

1. Y. Bando, T. Kuratate, and T. Nishita. “A simple method for modeling wrinkles on human skin”, *Proc. Pacific Graphics 2002,* pp. 166-175, 2002.
2. J. Blinn. “Simulation of wrinkled surface”, *Proc. SIGGRAPH’78,* pp. 286-292, 1978.
3. L. Boissieux, G. Kiss, N. M. Thalmann, and P. Kalra. “Simulation of skin aging and wrinkles with cosmetics insight”, *Proc. Eurographics Workshop on ComputerAnimation and Simulation 2000,* pp. 15-27, 2000.
4. S. Hadap, E. Bangerter, P. Volino and N. M. Thalmann. “Animating wrinkles on clothes”, *Proc. IEEE Visualization’99,* pp. 175-182, 1999.
5. G. Miller. “The motion dynamics of snakes and worms”, *Proc. SIGGRAPH’88,* vol.22, pp. 169-178, 1988.
6. M. Nahas, H. Huitric, M. Rioux, and J. Domey, “Facial image synthesis using skin texture recording”, *The Visual Computer,* 6(6): 337-343, 1990.
7. D. Terzopoulus and K. Waters. “Physically-based facial modeling, analysis and animation”, *Journal of Visualization and Computer Animation,* vol.1, pp. 73-80, 1990.
8. M. L. Viaud and H. Yahia. “Facial animation with wrinkles”, *Proc. Eurographics Workshop on Animation and Simulation’92,* pp. 1-13, 1992.
9. P. Volino and N. M. Thalmann. “Fast geometrical wrinkles on animated surfaces”, *Proc. WSCG’99,* 1999.
10. K. Waters. “A muscle model for animating three-dimensional facial expression”, *Proc. SIGGRAPH’87,* vol.21, pp. 17-24, July 1987.
11. Y. Wu, P. Kalra, L. Moccozet, and N. M. Thalmann. “Simulating wrinkles and skin aging”, *The Visual Computer,* 15(4): 183-198, 1999.
12. Y. Wu, P. Kalra, and N. M. Thalmann. “Physically-based wrinkle simulation & skin rendering”, *Proc. Eurographics Workshop on Animation and Simulation’97,* pp. 69-79, 1997.
13. Y. Zhang, E. C. Prakash, and E. Sung. “Efficient modeling of an anatomy-based face and fast 3D facial expression synthesis.” *Computer Graphics Forum*, 22(2): 159-169, June 2003.

A Multiresolutional Approach for Facial Motion Retargetting Using Subdivision Wavelets

Kyungha Min and Moon-Ryul Jung*

Dept. of Media Technology,
Graduate School of Media Communications, Sogang Univ., Seoul, Korea
{minkh, moon}@sogang.ac.kr

Abstract. We propose a new facial motion retargetting method using wavelet-based multiresolutional analysis of triangular meshes. We define the displacement of the source model as the difference between the neutral and the current expressions of the source model, and decompose it to the displacement of the base mesh and the "displacements" of the wavelet coefficients. We compute the wavelet coefficients of a mesh by using the wavelet basis functions derived from a mesh subdivision scheme. To get the motion of the target model, we add the displacement of the source model to the target model level by level starting from the base mesh. The wavelet-based representation of mesh facilitates the comparison between two expressions of the same model and the comparison between the expressions of the source and target models.

1 Introduction

Digital facial animation is one of the most popular research areas in computer graphics. The development of entertainment markets such as computer games, animations and movies accelerates the progress of digital facial animation techniques to build an efficient framework of realistic facial animation. In human body animation, motion retargetting, which creates new motions of the target model from the motion data of the source model, has introduced an innovative tool for creating realistic character animation [3]. The major difficulty of applying motion retargetting to facial models is that faces do not have the obvious control parameters to represent and control facial motions such as joint angles for body motion [7].

The problem of motion retargetting can be formulated as the problem of obtaining the current expression of the target model by adding the scaled version of the displacement of the source model to the neutral expression of the target model(See Fig. 1). To handle the problem of facial motion retargetting, we propose a multiresolution approach. To materialize this approach, we present two key ideas.

* This research has been supported by the fund (R01-2002-000-00311-01) from KOSEF (Korea Science and Engineering Foundation).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 216–223, 2005.
© Springer-Verlag Berlin Heidelberg 2005

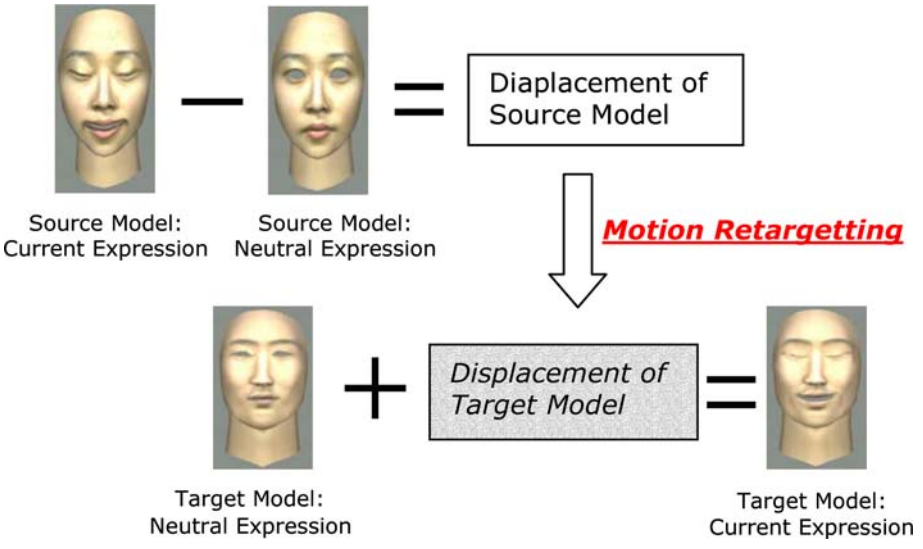


Fig. 1. The process of motion retargetting: Add the scaled version of the displacement of the source model to the neutral expression of the target model to get the current expression of the target model

The first key idea is that we retarget the displacement of the base mesh of the source model by using **the displacement parameters with respect to the local coordinate system defined by the normal vector and the principal directions on the base mesh**. Since the principal directions are good at representing the local geometry and orthogonal to each other [1], we use the normal vector and the two principal directions to define the local coordinate systems. It contrasts to the approach of Noh and Nuemann [6], where the source model is deformed to the target model to define the local coordinate systems for the target model, which correspond to those on the source model.

The second key idea is to represent the facial mesh as **a multi-resolution mesh and use the wavelet coefficients as the "control parameters" of the facial motion**. To represent the source facial model faithfully, we decompose the model into the base model and wavelet coefficients that store the details of the model using a multiresolution scheme called "subdivision wavelets" [4]. We decompose the displacement of the source model into the displacement of the base model and those of the wavelet coefficients. We then retarget the multi-level displacements to the target model.

2 Previous Work

Retargetting motions to new bodies were first developed by Gleicher[3] to make captured motions reusable. This idea was also applied to facial motion, and

there are some work[5, 6, 8, 10] in facial motion retargetting. Noh and Neumann [6] compute the displacement of each vertex on the source mesh at each frame relative to the neutral frame, and add the scaled version of the displacements to the neutral frame of the target mesh. If there are frames whose displacements from the neutral frame are large, the retargetted version of the displacements tend to intersect each other. Noh and Neumann's method has to apply filtering to the retargetted displacements to remove the intersections.

Pyun and Shin [8] proposed an example-based retargetting scheme. In this approach, the system uses a set of correspondences between example poses of the source and target facial models provided by the animator. This method produces more accurate results than that of Noh and Neumann, because of sufficient mapping data between source and target expressions. However, the more detailed the facial model is, the larger number of example expressions are needed.

Na and Jung[5] represent each frame of the source mesh by the base mesh and a sequence of normal offsets from the base mesh. The displacement of the source base mesh is retargetted by using an example-based approach of [8]. The fine motion of the target model is obtained by adding the normal offset displacements of the source mesh to the normal offsets of the target mesh at the neutral frame, level by level.

Wang et al. [10] learn a decomposable generative model from a set of facial motions of different people. A given motion of the source face is first decomposed into the content (e.g. smile) and the style parameters (a particular style of smile). Then the content motion of the source is transformed to the target motion by applying the scale factor between the source and target faces and the style parameters to the decomposable generative model.

3 Multiresolution Representation of Meshes

3.1 Construction of the Facial Models

The facial models used in this paper are triangular meshes with subdivision connectivity created by an experienced designer. Given a mesh without subdivision connectivity, we can remesh it so that it has subdivision connectivity by using the method proposed by Eck et al. [2].

3.2 Mesh Representation Using Subdivision Wavelets

For our experimentation we apply multi-resolution analysis to a given mesh three times to obtain the wavelet coefficients for three levels of the mesh. More levels of wavelet coefficients can be used to better approximate given meshes.

To represent a function using wavelet coefficients, we need to have appropriate basis functions, that is, scaling functions and wavelet functions. As Lounsbery and his colleagues [4] have shown that appropriate scaling and wavelet functions can be constructed for a mesh created by means of a given subdivision scheme, we construct appropriate scaling and wavelet functions to represent the mesh.

The wavelet functions constructed this way are called "subdivision wavelets". In this paper, we use the Butterfly subdivision scheme, an interpolation subdivision scheme, since it facilitates the computation of the analysis and synthesis filters. The analysis filters A^j, B^j and synthesis filters P^j, Q^j are determined by the subdivision scheme employed and the scaling functions derived from the subdivision scheme.

As shown in Figure 3.2, analysis filter A^j filters a given mesh to the next lower resolution. Analysis filter B^j computes v^j, the wavelet coefficients, which is the difference between the consecutive resolutions.

The filters A^j, B^j are represented as matrices, as follows:

$$\mathsf{V}^j \;=\; A^j \mathsf{V}^{j+1} \qquad\qquad \mathsf{v}^j \;=\; B^j \mathsf{V}^{j+1}.$$

Given V^j and v^j, V^{j+1} is reconstructed by applying the synthesis filters P^j and Q^j as follows:

$$\mathsf{V}^{j+1} \;=\; P^j \mathsf{V}^j + Q^j \mathsf{v}^j$$

In our subdivision scheme, P^j and Q^j are computed as follows:

$$(P^j \quad Q^j) \;=\; \begin{pmatrix} A^j \\ B^j \end{pmatrix}^{-1} \;=\; \begin{pmatrix} \mathbf{I}^j & -\alpha^j \\ \mathbf{N}^j & \mathbf{I}^j \;-\; \mathbf{N}^j \alpha^j \end{pmatrix} \tag{1}$$

Here $\mathbf{N^j}$ represents the portion of the subdivision matrix which weight the "new vertices" introduced by each application of subdivision. $\mathbf{I^j}$ is the portion of the subdivision matrix which weight the "old vertices", which is the identity matrix, because the Butterfly subdivision is an interpolation scheme. The matrix α^j are used to define the wavelet functions in terms of the scaling functions, and are computed by solving a linear system involving the inner products of the scaling functions. See Lounsbery and et al. [4] for details.

Given the base mesh V^0, a series of wavelet coefficients v^j, and the synthesis filters P^j, Q^j, the original mesh V^n is reconstructed by the following formula:

$$\mathsf{V}^n \;=\; (\prod_{i=0}^{n-1} P^i)\mathsf{V}^0 \;+\; \sum_{j=0}^{n-1} (\prod_{k=j+1}^{n-1} P^k) Q^j \mathsf{v}^j \tag{2}$$

Let the neutral expression and the current expression of the target model be $\mathsf{T_0}$ and $\mathsf{T_j}$. They have the following representations:

$$\mathsf{T_0} \;=\; (\; \mathsf{T}_0^0,\; \mathsf{t}_0^0,\; \mathsf{t}_0^1,\; \mathsf{t}_0^2 \;) \qquad\qquad \mathsf{T_j} \;=\; (\; \mathsf{T}_j^0,\; \mathsf{t}_j^0,\; \mathsf{t}_j^1,\; \mathsf{t}_j^2 \;)$$

The motion retargetting is achieved by adding the displacement between $\mathsf{S_0}$ and $\mathsf{S_j}$ to $\mathsf{T_0}$ in order to obtain $\mathsf{T_j}$.

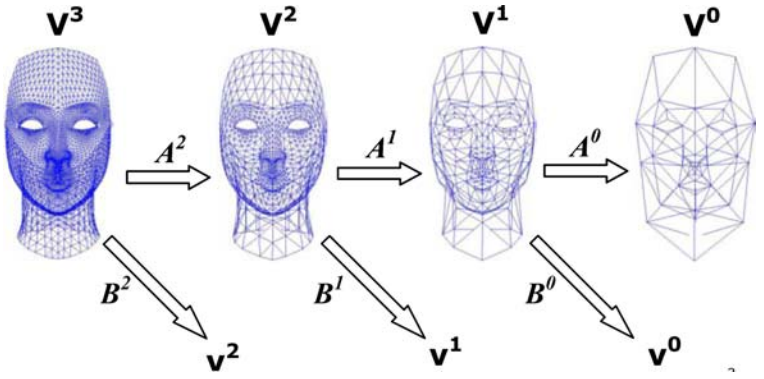


Fig. 2. Multiresolutional analysis of a facial model: The original model V^3 is filtered down to the base mesh V^0, and the wavelet coefficients, v^j, are computed in the filtering process

3.3 The Motion Retargetting Procedure

The multiresolutional motion retargetting is achieved in the following steps.

Step 1. Compute the displacement $\mathsf{M_j}$ of the source model between the neutral expression and the current expression, as follows:

$$\mathsf{M_j} = (\ \mathsf{M_j^0},\ \mathsf{m_j^0},\ \mathsf{m_j^1},\ \mathsf{m_j^2}\ ),\ \text{where}$$

$$\mathsf{M_j^0} \equiv \mathsf{S_j^0} - \mathsf{S_j^0},\ \text{and}\ \mathsf{m_j^i} \equiv \mathsf{s_j^i} - \mathsf{s_0^i},\ \text{for}\ 0 \le i \le 2. \tag{3}$$

The displacement of the base mesh $\mathsf{M_j^0}$ is defined as the difference of the current expression $\mathsf{S_j^0}$ from the neutral expression $\mathsf{S_0^0}$, as follows:

$$\mathsf{M_j^0}\ (\equiv \mathsf{S_j^0} - \mathsf{S_0^0}\ ) = \{\mathbf{d_i} \mid \mathbf{d_i} = \mathbf{p_j^i} - \mathbf{p_0^i},\ \text{where}\ \mathbf{p_j^i} \in \mathsf{S_j^0}\ \text{and}\ \mathbf{p_0^i} \in \mathsf{S_0^0}\},$$

where $\mathbf{p_0^i}$ and $\mathbf{p_j^i}$ are the corresponding vertices between the base mesh of the neutral expression and the base mesh of the current expression.

As shown in Figure 3, the displacement vector $\mathbf{d}_i$ is represented as follows:

$$\mathbf{d}_i = \alpha_i \mathbf{N}_i + \beta_i \mathbf{J}_i + \gamma_i \mathbf{K}_i \tag{4}$$

Here $\mathbf{N_i}$ is the normal vector, and $\mathbf{J_i}$ and $\mathbf{K_i}$ are the principal directions at the location $\mathbf{p_0^i}$. The principal directions of a triangular mesh are computed by the algorithm in [9].

As shown in Eqn 3, the displacement of upper levels of the mesh, $\mathsf{m_j^i}$, is defined to be the difference of the wavelet coefficients between the neutral expression and the target expression. The wavelet coefficients can be compared and thus subtracted because the wavelet basis functions on the neutral expression and the target expression correspond to each other.

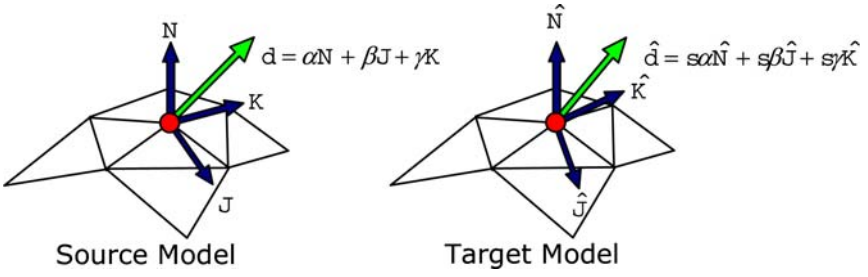


Fig. 3. Retargetting the displacement $\mathbf{d_i}$ on the source base mesh to the displacement $\hat{\mathbf{d}}_\mathbf{i}$ on the target base mesh

Step 2. Transform the displacement of the source model, (M_j^0, m_j^0, m_j^1, m_j^2) to the displacement of the target model, ($\hat{\mathsf{M}}_\mathsf{j}^0$, $\hat{\mathsf{m}}_\mathsf{j}^0$, $\hat{\mathsf{m}}_\mathsf{j}^1$, $\hat{\mathsf{m}}_\mathsf{j}^2$).

Each motion vector $\hat{\mathbf{d}}_i$ in $\hat{\mathsf{M}}_\mathsf{j}^0$ is computed as follows:

$$\hat{\mathbf{d}}_i \;=\; s\alpha_i\hat{\mathbf{N}}_i + s\beta_i\hat{\mathbf{J}}_i + s\gamma_i\hat{\mathbf{K}}_i \tag{5}$$

as shown in the right of Figure 3. Here ($\hat{\mathbf{N}}_\mathbf{i}$, $\hat{\mathbf{J}}_\mathbf{i}$, $\hat{\mathbf{K}}_\mathbf{i}$) is the target local coordinate system defined at the corresponding target vertex. Here s is the scale factor, which is defined to the ratio between the surface area of the source base mesh and that of the target mesh mesh. We use $\hat{\mathsf{m}}_\mathsf{j}^\mathsf{i} = \mathsf{m}_\mathsf{j}^\mathsf{i}$. In other words, we use the displacements of the wavelet coefficients on the source model as the displacements of the wavelet coefficients on the target model, for the following reasons. The wavelet functions for each level of the target model have the same forms as those of the source model. Though the supports of the wavelet functions are not the same between the source model and the target model, these supports correspond to each other between the source and target models, starting from the base mesh.

Step 3. Obtain the current expression of the target model T_j = (T_j^0, t_j^0, t_j^1, t_j^2) by adding the displacement of the target model computed in **Step 2** to the neutral expression T_0 = (T_0^0, t_0^0, t_0^1, t_0^2). That is, compute:

$$\mathsf{T}_\mathsf{j} \;=\; (\; \mathsf{T}_\mathsf{j}^0,\; \mathsf{t}_\mathsf{j}^0,\; \mathsf{t}_\mathsf{j}^1,\; \mathsf{t}_\mathsf{j}^2\;),\ \text{where}$$

$$\mathsf{T}_\mathsf{j}^0 \;=\; \mathsf{T}_0^0 \;+\; \hat{\mathsf{M}}_\mathsf{j}^0,\ \text{and}\ \mathsf{t}_\mathsf{j}^\mathsf{i} \;=\; \mathsf{t}_0^\mathsf{i} \;+\; \hat{\mathsf{m}}_\mathsf{j}^\mathsf{i},\ \text{for}\ 0 \;\leq\; i \;\leq\; 2.$$

Step 4. Finally, reconstruct the mesh corresponding to T_j by using Eqn 2:

$$\mathsf{mesh}(\mathsf{T}_\mathsf{j}) \;=\; P^2P^1P^0{\mathsf{T}_\mathsf{j}}^0 + P^2P^1Q^0{\mathsf{t}_\mathsf{j}}^0 + P^2Q^1{\mathsf{t}_\mathsf{j}}^1 \;+\; Q^2{\mathsf{t}_\mathsf{j}}^2.$$

4 Implementation and Results

We implemented the proposed algorithm in a personal computer with 2.0 GHz Pentium-4 CPU and 512 MByte Main Memory. Each of the facial models has

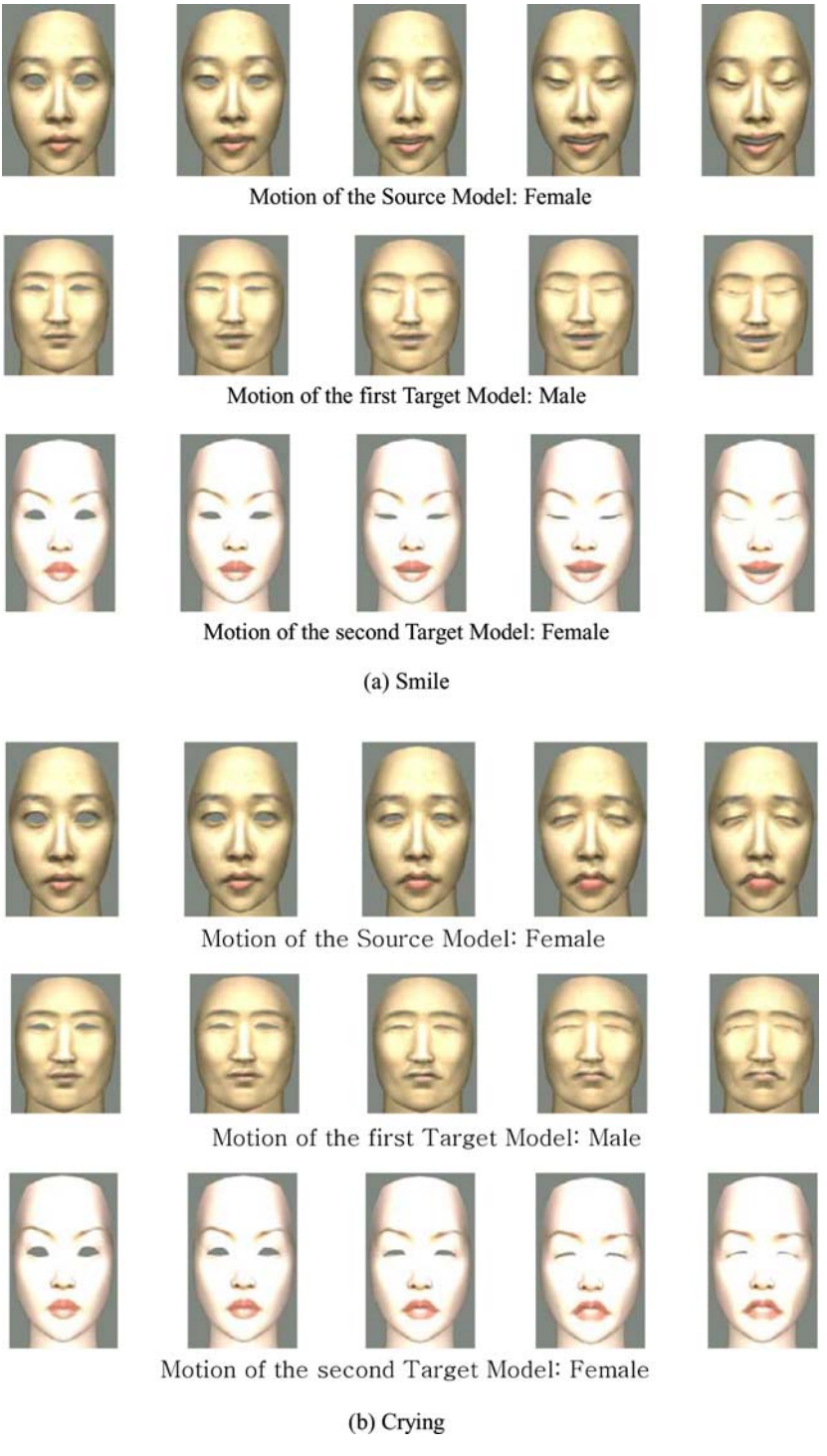


Fig. 4. Retargetting of the facial motions

2964 vertices and 5760 triangles. We tested the proposed algorithm for one source model and two target models. The source model is created from one female volunteer, and two target models are created from another female volunteer and one male volunteer. The motions we tested in this paper are smiling (See Fig. 4 (a)) and crying (See Fig. 4 (b)).

5 Conclusion

In this paper, we presented a new facial motion retargetting scheme based on the multiresolutional analysis of polygonal meshes. The displacement between the neutral and the current expressions are extracted from the base mesh of the source model and as well as from the wavelet coefficients. The algorithm creates the motion of the target model by adding the multiresolution displacements of the source model to the target base mesh and the wavelet coefficients of the target model at the neutral frame, level by level. In this process, the wavelet coefficients play the role of control parameters to represent and control facial motion. Hence we solved one of the inherent difficulties in facial motion retargetting, which is the lack of obvious control parameters to control facial motion.

References

1. do Carmo, M. P., Differential Geometry of Curves and Surfaces, Pearson Education, 1976.
2. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W., "Multiresolution Analysis of Arbitrary Meshes", Proceedings of SIGGRAPH 1998 Annual Conference Series, pp. 173–182, 1995.
3. Gleicher, M., "Retargetting motion to new characters," Proceedings of SIGGRAPH 1998 Annual Conference Series, pp. 33-42, 1998.
4. Lounsbery, M., DeRose, T. D., and Warren, J., "Multiresolution analysis for surfaces of arbitrary topological type," ACM Transactions on Graphics, Vol. 16, No. 1, pp. 34-73, 1997.
5. Na, K. and Jung, M., "Hierarchical Retargetting of Fine Facial Motions," Proceedings of Eurographics 2004, pp. 687-695, 2004.
6. Noh, J., Neumann, U., "Expression cloning," Proceedings of SIGGRAPH 2001 Annual Conference Series, pp. 277-288, 2001.
7. Parke, F. I., Waters, W., Computer Facial Animation, AK Peters, 1996.
8. Pyun, H. and Shin, S. Y., "An example-based approach for facial expression cloning," In ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2003), PP. 167-176.
9. Taubin, G., "Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation," Proceedings of IEEE ICCV 1995, pp. 852 - 857, 1995.
10. Wang, Y., Huang, X., Lee, C., Zhang, S., and Li. Z., "High Resolution Acquisition, Learning and Transfer of Dynamic 3D Facial Expressions," Proceedings of Eurographics 2004, pp. 677-686, 2004.

New 3D Graphics Rendering Engine Architecture for Direct Tessellation of Spline Surfaces

Adrian Sfarti, Brian A. Barsky, Todd J. Kosloff, Egon Pasztor,
Alex Kozlowski, Eric Roman, and Alex Perelman

University of California, Berkeley

Abstract. In current 3D graphics architectures, the bus between the triangle server and the rendering engine GPU is clogged with triangle vertices and their many attributes (normal vectors, colors, texture coordinates). We develop a new 3D graphics architecture using data compression to unclog the bus between the triangle server and the rendering engine. The data compression is achieved by replacing the conventional idea of a GPU that renders triangles with a GPU that tessellates surface patches into triangles.

1 Introduction

The main goal of this paper is to develop a new graphics architecture that exploits faster arithmetic such that the CPU will serve parametric patches and the rendering engine (GPU) will triangulate these patches in real time.

The proposed architecture handles freeform parametric rational and non-rational spline surfaces such as Bézier and B-spline/NURBS. In comparison with current graphics architectures, which are based on transferring triangle vertices, the amount of data sent over the bus between the CPU and the GPU will be reduced by a factor that is linear in the number of triangles forming each surface patch. That is, if a surface patch is tessellated into n triangles, then the bus bandwidth required is 1/n of the bandwidth that would have been required by the conventional method of transferring triangle vertices. Since this approach stores control points of the surface patch instead of triangle vertices, the memory footprint will also be reduced by this same factor. For example, today's top of the line GPUs are capable of rendering about 500 million meshed triangles per second. At about 30 bytes/vertex (color, texture coordinates, geometry, normals) this results into a bus bandwidth requirement of about 15Gbytes/second while the fastest buses available have a peak bandwidth about 4 times lower.

Since today's Transformation Units are programmable processors, we envisage our implementation not as a silicon implementation but rather as reprogramming an already existent Transformation Unit inside an already existent GPU. Our simulations show that due to the reduced transformation effort (we replace the transformation of the triangle vertices with the transformation of the surface control points) we can easily reprogram one of the Transformation Units to act as a Tesselator Unit.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 224–231, 2005.
© Springer-Verlag Berlin Heidelberg 2005

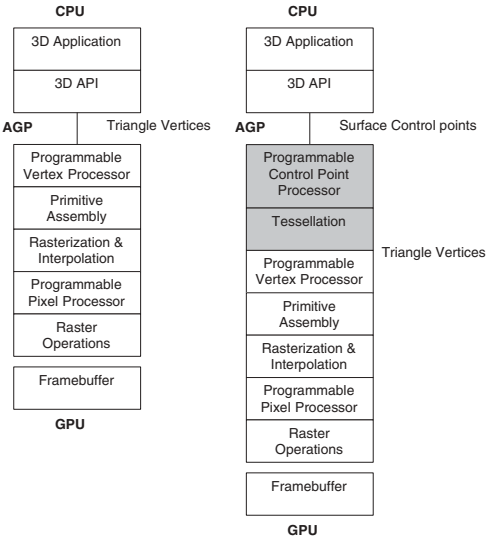


Fig. 1. Conventional programmable architecture (left), new architecture (right). The new architecture adds two stages to the GPU pipeline, which are shown in grey

2 Previous Work

There have been very few implementations of real time tesselation in hardware. In the mid-1980's, Sun developed an architecture for this that was described in [1] and in a series of associated patents. The implementation was not a significant technical or commercial success because it did not exploit triangle based rendering; instead it attempted to render the surfaces in a pixel-by-pixel manner [2]. The idea was to use adaptive forward differencing to interpolate infinitesimally close parallel cubic curves imbedded into the bicubic surface.

More recently, Henry Moreton from NVIDIA has resurrected the real time tesselation unit [3]. This method does not directly tesselate patches in real time; rather, it uses off-line pre-tesselated triangle meshes in conjunction with a proprietary stitching method that avoids cracking and popping at the seams. Using this approach, the tesselation unit exists in front of the transformation unit and outputs triangle databases to be rendered by the existent components of the 3D graphics hardware.

The current paper is based on a recent patent [4] that is the first to introduce a real time tesselation processor into a GPU pipeline. To date, there is no GPU built with a real time tesselator processor but we hope that the current article will spark the design of such a device.

3 GPU Architectures

3.1 Current State of the Art

The pseudo code describing the current state of the art GPU architectures is shown below for the bicubic case. Notation: Let (s_i, t_j) denote a pair of parameter values used

in patch parameterization, $V_{i,j}$ a vertex, and $N_{i,j}$ a vertex normal. Also, we denote texture coordinates by $u_{i,j}$ and $v_{i,j}$.

Step 1. (off-line). For each bicubic surface, subdivide the S and T intervals until each resultant four-sided surface is below a certain predetermined curvature value.

Step 2. For all bicubic surfaces sharing a common boundary, take the union of the subdivisions to prevent cracks along the common boundary.

Step 3. For each bicubic surface, For each pair (s_i,t_j), Calculate $(u_{i,j},v_{i,j},q_{i,j},V_{i,j})$, Generate triangles by connecting neighboring vertices.

Step 4. For each vertex $V_{i,j}$ Calculate the normal $N_{i,j}$ to that vertex (used for lighting). For each triangle Calculate the normal to the triangle (used for culling).

Step 5. (real time). Transform the vertices $V_{i,j}$ and the normals $N_{i,j}$ and the normals to the triangles. For each vertex $V_{i,j}$, Calculate lighting.

3.2 The Tesselator Unit – Principles of Operation

Though we have not implemented our proposed GPU in silicon yet we are publishing below the behavioral Verilog code describing the principles of operation. From this behavioral code a silicon implementation of the proposed GPU architecture should be straightforward:

Step 0 (all steps in real time). For each bicubic surface Transform the 16 control points that determine the surface.

Step 1. For each bicubic surface, Subdivide the boundary curve representing the s interval until the projection of the length of the height of the curve bounding box is below a certain predetermined number of pixels as measured in screen coordinates. Repeat the process for the boundary curve representing the t interval. For each new view, a new subdivision can be generated, producing automatic level of detail.

Step 2. For all bicubic surfaces sharing a same parameter (either s or t) boundary, Choose as the common subdivision the reunion of the subdivisions in order to prevent cracks showing along the common boundary OR choose as the common subdivision the finest subdivision (the one with the most points inside the set) OR insert a zippering strip to smoothly progress from one patch to its neighbor. (Prevent cracks at the boundary between surfaces).

Step 3. (Generate the vertices, normals, the texture coordinates and the displacements used for bump and displacement mapping for the present subdivision) For each bicubic surface, For each pair (s_i,t_j) Calculate $((u_{i,j},v_{i,j},q_{i,j}),(p_{i,j},r_{i,j}),V_{i,j})$ through interpolation (texture , displacement map and vertex coordinates as a function of (s_i,t_j)) Look up vertex displacement $(dx_{i,j},dy_{i,j},dz_{i,j})$. Generate triangles by connecting neighboring vertices.

Step 4. For each vertex $V_{i,j}$ Calculate the normal $N_{i,j}$ to that vertex.

4 The Subdivision Step

We use the Lane-Carpenter subdivision algorithm described in [5] but we apply our own termination criterion. The four boundary curves of a Bézier patch are themselves Bézier curves, which we subdivide using the de Casteljau formulas. The edge subdivision results in a subdivision of the parametric intervals These parameter values are stored, whereas the control points resulting from subdivision are discarded immediately after the termination test is run. After the subdivision and crack prevention steps, the actual vertex locations throughout the patch are computed from the stored parameter values. The generation of vertices is comparable with vertex transformation. Note that the vertices are generated already transformed in place because the parent bicubic surface has already been transformed.

5 Termination Criteria

Our algorithm decides that an edge curve has been sufficiently subdivided when the trapezoidal convex hull of that curve has a sufficiently small height, as seen from the viewpoint of the observer. Referring to Figure 2, subdivision terminates when the following conditions are met:

$$\text{Max}\{\text{dist}(P_{12}\text{to line}(P_{11},P_{14})),\text{dist}(P_{13}\text{to line}(P_{11},P_{14}))\} \times \frac{2d}{(|P_{12_z}|+|P_{13_z}|)} < n \text{ AND}$$
$$\text{Max}\{\text{dist}(P_{24}\text{to line}(P_{14},P_{44})),\text{dist}(P_{34}\text{to line}(P_{14},P_{44}))\} \times \frac{2d}{(|P_{24_z}|+|P_{34_z}|)} < n$$

where n is an arbitrary number expressed in pixels or in a fraction of pixels and d is the distance from the viewer to the projection plane.

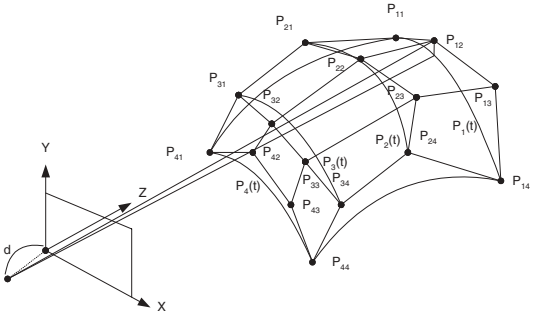


Fig. 2. Termination Criteria

We experimented with n starting at 1 and we observed that there were artifacts, especially along the silhouette. Forsey et al. [6] seem to settle on $n = .5$ and we tried that. We also experimented with $n > 1$, for reasons of rapid prototyping and previewing.

The above criterion is sufficient for surface patches that are not more curved inside their boundaries than they are along their boundaries. Conversely, if the patches are more curved inside than they are along their boundaries, we terminate subdivision based on the internal curves of the patch.

Since the curvature of free-form surfaces can switch between being boundary-limited and internally-limited, we will need to measure the flatness of both types of curves at the start of the tesselation associated with each instance of the surface by subdividing the four boundary curves as well as two orthogonal internal curves, specifically the curves that interverne in the second termination criterion shown above. We can further exploit the fact that adjacent patches share two boundary curves so that we need to subdivide only two of the four boundary curves for each patch. The only obvious exceptions are the patches at the boundary of a surface, since such patches have fewer than four neighbors. In the case of the boundary patches, our algorithm always subdivides all four boundary curves.

Our termination test ensures that patches are subdivided sufficiently to avoid silhouette artifacts. However, the test was shown to be insufficient in the case of a large flat patch facing the viewer. Such a patch would not be subdivided because a single pair of triangles can completely capture the geometry of this curve. However, per-vertex lighting would lead to highlight artifacts. Additionally, nearly-flat portions of a patch exhibit undesirable texture flickering as they visibly transition from one level of detail to the next. This is because the bilinear texture coordinate interpolation that we use when assigning texture coordinates to triangle vertices is not the same as the method used to interpolate within a triangle. To combat this problem, we decree that patches must be subdivided a minimum number of times, regardless of curvature.

6 Crack Prevention

If there are no special prevention methods, cracks may appear at the boundary between abutting patches. This is mainly due to the fact that the patches are subdivided independently of each other. Abutting patches can exhibit different curvatures resulting in different subdivisions. When rendering the parallel strips of triangles to the left and to the right of the common boundary, a crack may become visible.

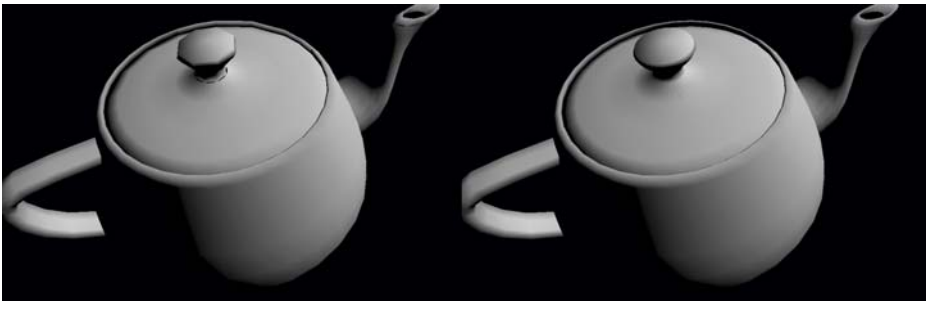


Fig. 3. Left: Without Crack Prevention. Right: With Crack Prevention.

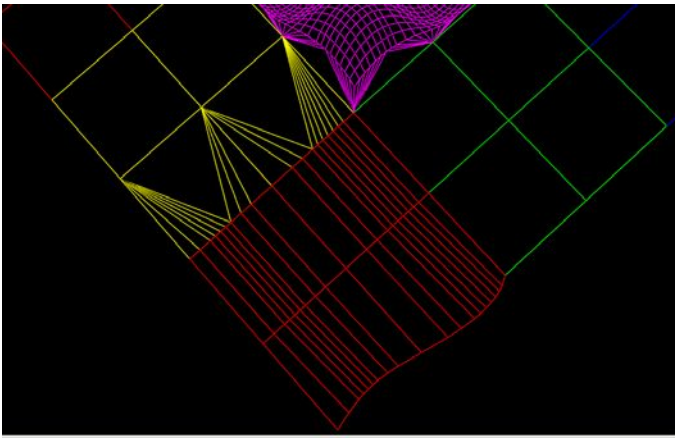


Fig. 4. Zippering In Action

We use zippering to prevent cracks from forming. Zippering leaves the interior of patches untouched, allowing these regions to be tessellated without concern for neighboring patches. To eliminate cracks between adjacent patches (as in Figure 3), the portion of a patch that is immediately in contact with an adjacent patch is carefully tessellated using a zipper-like configuration, so as to seamlessly move from a lower to higher level of tesselation. An example of zippering is shown in Figure 4. We tested the crack prevention algorithm on a large selection of objects, making sure that the method works on corner cases such as the fans of patches shown in figure 4.

7 Performance Measurements

In order to measure the performance of the algorithm, we tested our prototype on five dynamic scenes. Each scene comprised seven teapots that were spun along different elliptical paths. At any given moment, some teapots are close to the viewer, while others are far away.

All tests were performed on a Pentium 4 2.4Ghz machine with a GeForce4 MX440 video card. We realize that the performance testing is done on a software *simulation* of the Tesselator Unit architecture since none of the existent GPU's has one today. Therefore, the performance numbers are only indicative of the performance of the actual architecture. Nevertheless, it was immediately observed that the dynamic tessellation compares favorably with the fixed tesselation since it shows higher frame rates in most cases, as described below.

The quality of the rendering is improved as well , especially for the cases when the objects are very close to the viewer. We observed no disadvantages to our method as compared to the conventional method. We experimented with a scene that had a fixed tesselation of 64k triangles. By using the real time tesselation the number of triangles varied from a maximum of 16k (closest from the viewer) down to a few hundreds

RTT					
N	Triangles Generated	Transform + Tessellate		Transform + Tessellate + Render	
		milliseconds	fps	milliseconds	fps
0.5	29~34K	21.7	43.1	23.6	40.1
0.7	24~28K	19.9	47.0	21.5	43.8
1.0	24~26K	19.3	48.3	20.9	45.0
2.0	24~26K	19.2	48.7	20.8	45.3
Offline Tessellation					
Triangles Per Patch	Triangles Generated Overall	Transform		Transform + Render	
		milliseconds	fps	milliseconds	fps
128	28K	6.3	60+	15.2	60+
512	115K	18.3	51.3	30.1	32.0
2048	459K	66.7	14.5	81.9	11.9

Fig. 5. Performance Measurements: Averages

(farthest from the viewer) clearly demonstrating the compression capabilities of our method. Since the main performance savings in our architecture will come from the AGP/PCIX bus bandwidth reduction we wanted to verify that our approach does not create a bottleneck inside the GPU. We obtained ample proof that this is not the case by trying many scenarios that allow for complex animations of flocks of objects in the context of a variable viewpoint . We have made several movies of our animations as well as a menu driven interactive demo.

The "RTT" entry represents our real time tesselation method. We separated the measurements into transform+tesselate only vs. transform+tesselate+render. The reason is that we wanted to measure the exact effects of the tesselation by comparison with conventional offline tesselation. In a real GPU there would be a tesselator unit stage, so that the effects of tesselation on execution time would be hidden by the fact that the GPU is pipelined. We rendered the same animation four different times, each time at a different criterion of subdivision termination ($n = 0.5$, $n = 0.7$, $n = 1$, $n = 2$). As n (the fractional deviation of the planar approximation from the real surface, expressed in pixels) increases, the number of triangles generated from subdivision decreases and the speed increases.

We also implemented a feature to simulate the current standard rendering methods whereby models are tessellated offline and then sent to the GPU as sets of triangles. This is the "Offline Tesselation" entry at the bottom of the table. Each patch is tessellated uniformly to a user-defined number of triangles (128, 512, or 2048). On every frame, each pre-calculated vertex is transformed from model space into world coordinates. The normal of each vertex is also appropriately transformed into world coordinates. Then the triangle is rendered directly.

8 Conclusion

We developed a new 3D graphics architecture using data compression to unclog the bus between the triangle server and the rendering engine. The data compression is achieved

by replacing the conventional idea of a rendering engine that renders triangles with a rendering engine that will tessellate surface patches into triangles. Thus, the bus sends control points of the surface patches, instead of the many triangle vertices forming the surface, to the rendering engine.

References

1. Lien, S.L., Shantz, M., Pratt, V.R.: Adaptive forward differencing for rendering curves and surfaces. In: SIGGRAPH '87 Proceedings, ACM (1987) 111–118
2. Lien, S.L., Shantz, M.: Shading bicubic patches. In: SIGGRAPH '87 Proceedings, ACM (1987) 189–196
3. Moreton, H.P.: Integrated tesselator in a graphics processing unit. U.S. patent (2003) #6,597,356.
4. Sfarti, A.: Bicubic surface rendering. U.S. patent (2003) #6,563,501.
5. Lane, J.F., Carpenter, L.C., Whitted, J.T., Blinn, J.F.: Scan line methods for displaying parametrically defined surfaces. In: Communications of the ACM. Volume 23(1)., ACM (1980) 23–24
6. Forsey, D.R., Klassen, R.V.: An adaptive subdivision algorithm for crack prevention in the display of parametric surfaces. In: Proceedings of Graphics Interface. (1990) 1–8
7. Clark, J.H.: A fast algorithm for rendering parametric surfaces. In: Computer Graphics (SIGGRAPH '79 Proceedings). Volume 13(2) Special Issue., ACM (1979) 7–12
8. Moreton, H.P.: Watertight tesellation using forward differencing. In: Proceedings of the ACM SIGGRAPH/Eurographcs workshop on graphics hardware. (2001)
9. Chung, A.J., Field, A.: A simple recursive tesselator for adaptive surface triangulation. JGT **5(3)** (2000)
10. Moule, K., McCool, M.: Efficient bounded adaptive tesselation of displacement maps. In: Graphics Interface 2002. (2002)
11. Boo, M., Amor, M., Dogget, M., Hirche, J., Strasser, W.: Hardware support for adaptive subdivision surface rendering. In: Proceedings of the ACM SIGGRAPH/Eurographics workshop on Graphics Hardware. (2001) 33–40
12. Hoppe, H.: View-dependent refinement of progressive meshes. In: Proceedings of the 24th annual conference on computer graphics and interactive techniques. (1997)
13. Sfarti, A.: System and method for adjusting pixel parameters by subpixel positioning. U.S. patent (2001) #6,219,070.
14. Barsky, B.A., DeRose, T.D., Dippe, M.D.: An adaptive subdivision method with crack prevention for rendering beta-spline objects. Technical Report, UCB/CSD 87/384, Computer Science Division, Electrical Engineering and Computer Sciences Department, University of California, Berkeley, California, USA (1987)
15. Velho, L., de Figueiredo, L.H., Gomes, J.: A unified approach for hierarchical adaptive tesselation of surfaces. In: ACM Transactions on Graphics. Volume 18(4)., ACM (1999) 329–360
16. Kahlesz, F., Balazs, A., Klein, R.: Nurbs rendering in opensg plus. In: OpenSG 2002 Papers. (2002)

Fast Water Animation Using the Wave Equation with Damping

Y. Nishidate and G. P. Nikishkov

University of Aizu, Aizu-Wakamatsu 965-8580, Japan
niki@u-aizu.ac.jp
http://www.u-aizu.ac.jp/~niki

Abstract. A simple method for animation of water waves is presented. The two-dimensional wave equation with damping is used to obtain a finite difference scheme for height distribution. A computational procedure employs explicit time integration. High frame rates are typically obtained for real-time animation of water waves.

1 Introduction

During the last two decades, researchers has devoted some attention to animation of water movement under various conditions. Detailed reviews of techniques for realistic water modeling and rendering have been presented by Iglesias [1] and Adabala and Manohar [2]. According to the reviews, approaches to dynamic modeling of fluids can be divided into visually convincing techniques and physically accurate techniques.

Visually convincing techniques usually use heuristic approaches to simulate fluid-like behavior, which creates 'nice' visual impression. These techniques can provide high frame rates for animation. However, they may require complicated tuning of empirical parameters.

Physically accurate techniques are mostly based on the numerical solution of some form of the Navier-Stokes equations [3, 4, 5, 6]. Three-dimensional problems have been solved by Foster and Metaxas [4]. Other authors [3, 5, 6] employ simplified two-dimensional Navier-Stokes differential equations. Since the solution of the Navier-Stokes equations is a computationally intensive task it usually makes real-time animation impossible. In computer graphics it is recognized that faster animation is more important than its physically-based realism. We devote our efforts to short turnaround time of water wave animation while still preserving simplified physical and mathematical foundations of an algorithm.

In this paper, we propose a method for animation of water waves, which is based on the two-dimensional wave equation with damping. It can be shown that the wave equation without damping is equivalent to the linearized shallow water equation. This justifies the use of the wave equation for modeling water waves. Using the wave equation with damping allows us to produce acceptable realistic picture of waves characterized by decreasing of wave height with time. Our dynamic model of water waves propagation utilizes explicit time integration

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 232–239, 2005.
© Springer-Verlag Berlin Heidelberg 2005

of discrete analogue of the wave equation with damping. Initial conditions are specified in the form of height excitation at one or several places. Additional excitations can be introduced at any time during animation procedure. This helps to simulate different phenomena such as rain drops or boat movement on the water surface. A computer code for animation of water waves typically provides high frame rates.

2 Computational Model

2.1 Governing equation

The linearized shallow water equation [6] assumes that water speed varies slowly in space:

$$\frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} = 0, \quad \frac{\partial v}{\partial t} + g\frac{\partial h}{\partial y} = 0, \quad \frac{\partial h}{\partial t} + d\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0\,. \tag{1}$$

Here it is supposed that the water surface is parallel to xy coordinate plane; u, v are fluid velocities along $x-$ and $y-$ directions; h is the height of the water surface; d is the water depth and g is the gravitational acceleration.

If to differentiate first equation in respect to x, second equation in respect to y, third equation in respect to t and to combine all three equations, the following second order differential equation containing only the height h can be obtained:

$$\frac{\partial^2 h}{\partial t^2} = gd\left(\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2}\right)\,. \tag{2}$$

It can be easily seen that for $d = const$ the above shallow water equation coincides with the two-dimensional wave equation without damping. From everyday experience we understand that the wave equation without damping is not sufficient for realistic water movement modeling. For example, wakes that spread behind a boat as it moves forward are decreased with time. This phenomenon can be modeled if to utilize the wave equation with damping, which can be written down in the following from:

$$\frac{\partial^2 h}{\partial t^2} - k\frac{\partial h}{\partial t} = c^2\left(\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2}\right)\,, \tag{3}$$

where c is the wave speed and k is the damping constant.

2.2 Finite difference procedure

Finite Difference Equation. For water surface animation, let us introduce a rectangular grid of nodal points for controlling the water height h. Each nodal point is characterized by two subscripts as shown in Fig. 1: i, which counts grid points along x and j, which counts grid points along y. Superscripts are used to relate the quantity to time moment. The water height h at location x_i, y_j at time moment t is denoted as $h^t_{i,j}$.

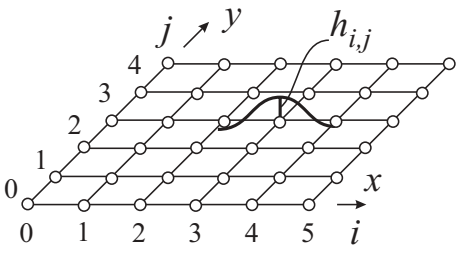


Fig. 1. Finite-difference grid for modeling wave movement

The finite difference approximation produces the following discrete analogue of Eq. (3) for equal point steps along coordinate axes $\Delta x = \Delta y$:

$$\frac{h_{i,j}^{t+\Delta t} - 2h_{i,j}^{t} + h_{i,j}^{t-\Delta t}}{\Delta t^2} + k\frac{h_{i,j}^{t} - h_{i,j}^{t-\Delta t}}{\Delta t} = c^2 \frac{4h_{i,j}^{t} - h_{i+1,j}^{t} - h_{i-1,j}^{t} - h_{i,j+1}^{t} - h_{i,j-1}^{t}}{\Delta x^2} \,. \tag{4}$$

After regrouping terms we arrive at the following expression for time integration of the water height:

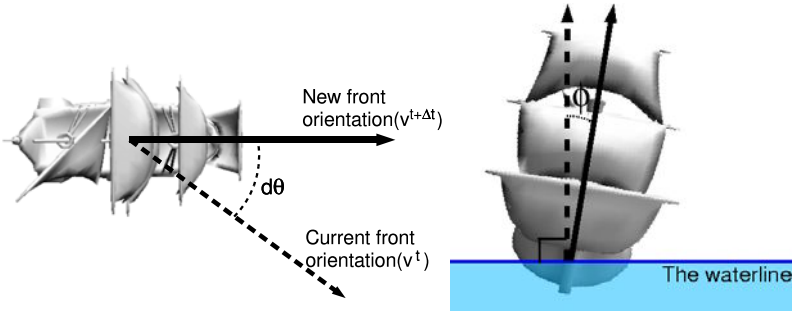
$$h_{i,j}^{t+\Delta t} = h_{i,j}^{t} + (1 - k\Delta t)\left(h_{i,j}^{t} - h_{i,j}^{t-\Delta t}\right) + \frac{\Delta t^2 c^2}{\Delta x^2}\left(4h_{i,j}^{t} - h_{i+1,j}^{t} - h_{i-1,j}^{t} - h_{i,j+1}^{t} - h_{i,j-1}^{t}\right) \,. \tag{5}$$

Influence of neighbors at time t in Eq. (5) is estimated with the use of the water height at four neighboring points: left, right, top and bottom. It is possible to take into account all eight neighboring points for the height computation at point i, j. Then the modified expression for time integration becomes:

$$h_{i,j}^{t+\Delta t} = h_{i,j}^{t} + (1 - k\Delta t)\left(h_{i,j}^{t} - h_{i,j}^{t-\Delta t}\right) + \frac{\Delta t^2 c^2}{\Delta x^2}[4h_{i,j}^{t} - h_{i+1,j}^{t} - h_{i-1,j}^{t} - h_{i,j+1}^{t} - h_{i,j-1}^{t} + \frac{1}{2}\left(4h_{i,j}^{t} - h_{i+1,j+1}^{t} - h_{i+1,j-1}^{t} - h_{i-1,j+1}^{t} - h_{i-1,j-1}^{t}\right)] \,. \tag{6}$$

Numerical experiments demonstrate that wave shapes look visually better when eight neighbors are taken into account. Thus, the Eq.(6) is preferable for water animation.

Integration, Initial Conditions and Boundary Conditions. An explicit scheme is used here for for time integration of the water profile. It has the obvious advantage of simplicity. A possible drawback of explicit integration is its instability for large integration time steps. The Courant-Friedrichs-Levy (CFL) stability condition requires that no wave travels farther than one cell in one time step. Our experience shows that for water animation problems the integration (5) or (6) is stable for real-time animation with reasonable values of the wave speed c and the damping constant k.



(a) Boat view from above (b) Boat view from front

Fig. 2. Boat orientation: the direction angle θ, increment of the direction angle $d\theta$ and the tilt angle ϕ

In order to compute the height values at time $t+\Delta t$ we need the height values for two previous time moments t and $t-\Delta t$. The initial conditions consist of: 1) setting $h^0_{i,j} = 0$ at time $t_0 = 0$; 2) setting of initial excitations $h^1_{i,j} = e_{i,j}$ at time $t_1 = \Delta t$. Nonzero height values at time t_1 can be specified at any number of grid points. Minimum initial excitation at time t_1 can include just one nonzero height value. After specification of two initial fields $h_{i,j}$, integration of Eq. (6) produces the water height fields for any time in the future. If the damping constant k is nonzero then the wave amplitude is decreasing with time.

In addition to initial excitation, any height values can be changed at any time moment. Using random number generators in space and in time, it is possible to simulate rain making indentations at the water surface, which imitate rain drops. Surface profile correction allows one to simulate movement of a boat on the water surface. Reflections of water waves at the boundary of the water reservoir are modeled with the symmetry boundary conditions. If the grid point i, j is located on the boundary then symmetrical interior points are employed instead of lacking height values outside the grid in the time integration procedure.

2.3 Boat on the Water Surface

Presented here the water wave animation procedure may be suitable for game development. Let us consider an application example where the boat follows the mouse position in a manner similar to magnetic force interaction. Boat movement is controlled by an acceleration vector $\boldsymbol{a}$ and by tilt angle ϕ as shown in Fig. 2. Components of the acceleration vector are calculated using location of the boat (x_b, y_b) and location of the mouse (x_m, y_m):

$$a_x = \frac{x_m - x_b}{\sqrt{(x_m - x_b)^2 + (y_m - y_b)^2}}, \quad a_y = \frac{y_m - y_b}{\sqrt{(x_m - x_b)^2 + (y_m - y_b)^2}}\,. \tag{7}$$

The above relations imitate magnetic force interaction. The acceleration vector is used to calculate the velocity vector v^t of the boat and new boat position.

The angle between current boat orientation and the acceleration vector $d\theta$ can be calculated as follows:

$$d\theta = \arccos\left(\left(\frac{v^t}{|v^t|}\right) \bullet \left(\frac{v^{t+\Delta t}}{|v^{t+\Delta t}|}\right)\right) . \tag{8}$$

The initial boat orientation is given by the programmer or by the user. The direction angle θ defining boat orientation and the tilt angle ϕ are calculated as:

$$\theta^{t+\Delta t} = \theta^t + d\theta, \quad \phi = \kappa\, d\theta , \tag{9}$$

where κ is an appropriate constant in order to make boat movement natural. After moving the boat to new position, negative increment is applied to the water height value at the previous boat position. This imitates wakes that spread behind the boat.

3 Implementation

The computational procedure for animation of water waves can be implemented in the following steps:

1. Define a grid for water height computation.
2. Apply initial deflections to some grid points.
3. Compute new water height values using wave profiles at two previous time moments using Eq. (6).
4. Render the water surface.
5. If necessary apply height excitations at grid points (rain drops, boat movement *etc.*)
6. Go to step 3.

To produce realistic water surface images with reflection, surface texture coordinates are generated following the sphere mapping algorithm [7]. The algorithm provides a reflection vector and texture coordinates that produce suitable reflections on the water surface.

The wave animation algorithm has been programmed in C++ language using OpenGL library for rendering. Desktop computer with Intel Pentium 4 2.4GHz processor, 512MB RAM and NVIDIA GeForce4 graphics card was used to perform real time animations. The algorithm was tested on square grids of different sizes, with different number of initial deflections and with different values of the model parameters.

4 Examples

In this section, several examples of application of the proposed water wave animation algorithm to various water movements are presented. In the first example, the surface deflection is applied to water surface at three points. The surface

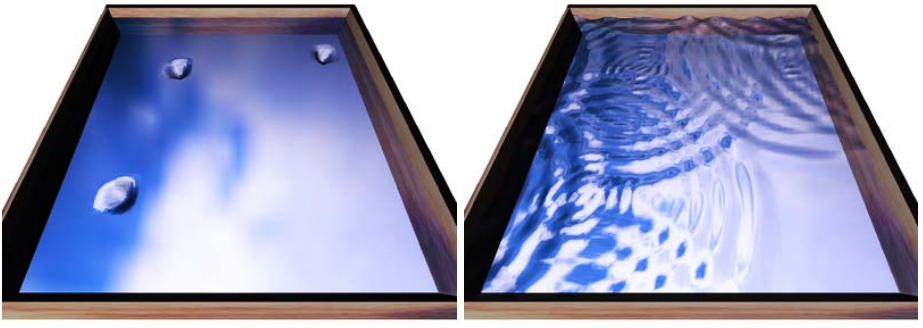


Fig. 3. Wave simulations with three initial deflections

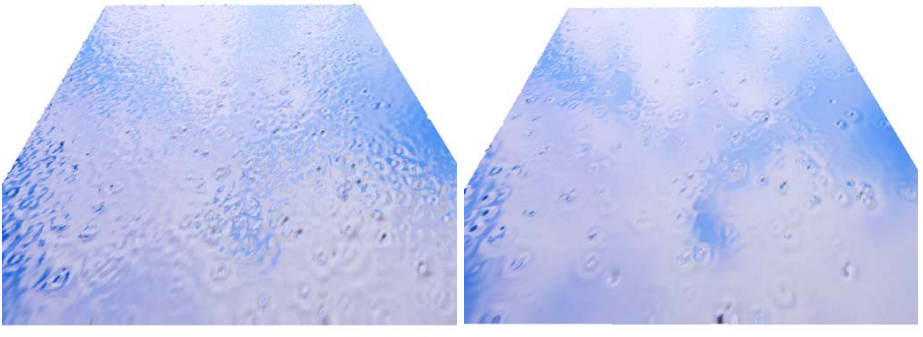


Fig. 4. Rain imitation with 128 drops: $R = 0.95$ (left), $R = 0.85$ (right)

profile after reflections of waves from the walls is shown in Fig. 3. The finite difference grid 64×64 is used. The frame rate is about 230 fps (frames per second). The value of the damping coefficient is set to zero in order to demonstrate effects of reflections and interference.

The example of Fig. 4 illustrates simulation of rain drops. The surface deflections are applied randomly at 128 grid points imitating rain drops. The surface is divided into the 128×128 grid. A coefficient $R = (1 - k\Delta t)$ is set to 0.95 and to 0.85 for the left and right pictures of Fig. 4 respectively. The code runs with the frame rate about 100 fps.

Fig. 5 shows simulation of wave propagation inside a concave-shaped reservoir. The Π-shaped water surface is produced by cutting off the part of the rectangular mesh. The surface is represented by the 64×64 grid. Four pictures demonstrate the process of spreading waves with reflections from the walls and interference. The frame rate is about 250.

Fig. 6 presents two screen shots of boat movement on the water surface. The coefficient $R = 0.95$ is employed. Fractal mountains are used for a coast line.

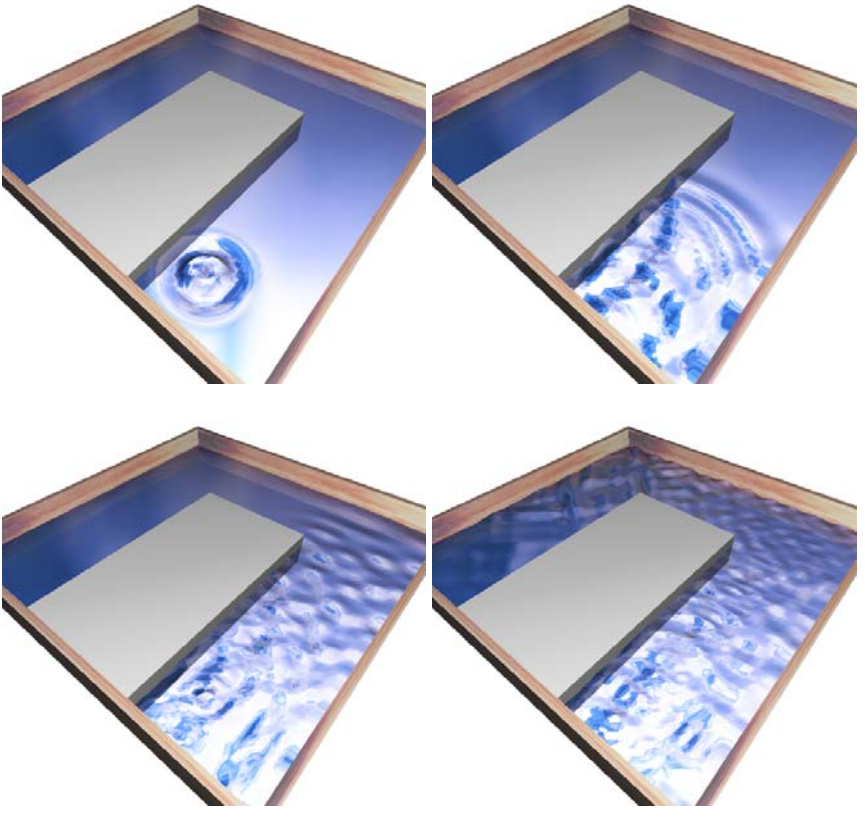


Fig. 5. Wave simulation applied to concave boundary surface

Boat trajectories and interference with some additional excitations can be seen in the pictures.

The reported frame rates are given for the code which performs both finite difference equation integration and water surface rendering for each frame. The algorithm efficiency can be increased if to perform rendering just with necessary for human eye frequency.

5 Conclusion

A simple and fast method for animation of water waves has been developed. Using the two-dimensional wave equation with damping, a finite difference relation for a water surface profile was obtained. An explicit scheme was used for time integration. One empirical coefficient was employed in order to simulate decrease of wave amplitudes with time.

Our experiments with the proposed water wave model show that it provides visually realistic animations of water waves with high frame rates. For example,

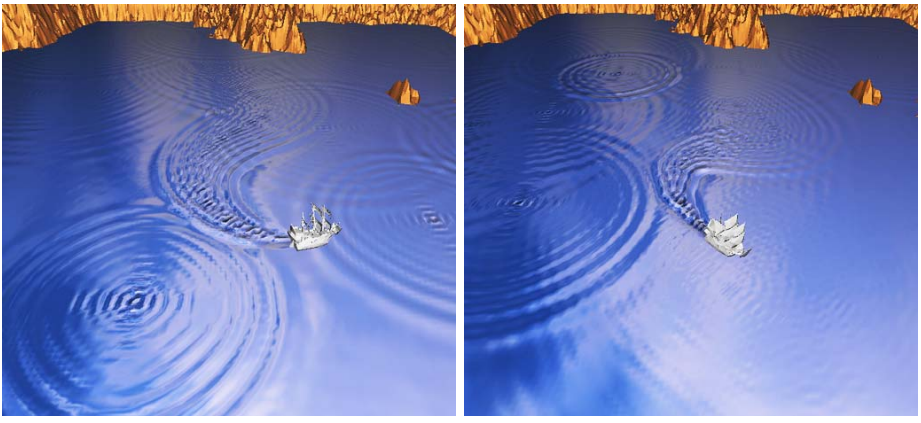


Fig. 6. Boat on the water surface

frame rate 100 fps was obtained for a mesh of 128 × 128 points using Intel Pentium 4 2.4GHz computer.

References

1. Iglesias, A.: Computer graphics for water modeling and rendering: a survey. Future Generation Computer Systems, **20** (2004) 1355-1374.
2. Adabala, L., Manohar, S.: Techniques for realistic visualization of fluids: a survey. Computer Graphics Forum **21** (2002) 65-81.
3. Kass, M., Miller, G.: Rapid, stable fluid dynamics for computer graphics. Computers and Graphics **24** (1990) 4955.
4. Foster, N., Metaxas, D.: Realistic animation of liquids. Graphical Models and Image Processing **58** (1996) 471483.
5. Chen, J.X., Lobo, N.V., Hughes, C.E., Moshell, J.M.: Real-time fluid simulation in a dynamic virtual environment. IEEE Computer Graphics and Applications. May-June 1997, 52-61.
6. Layton, A.T., van de Panne, M.: A numerically efficient and stable algorithm for animating water waves. The Visual Computer **18** (2002) 41-53.
7. Haeberli, P., Segal, M.: Texture mapping as a fundamental drawing primitive. Fourth Eurographics Workshop on Rendering, June 1993, 259-266.

A Comparative Study of Acceleration Techniques for Geometric Visualization

Pascual Castelló, José Francisco Ramos, and Miguel Chover

Department of Computer Languages and Systems, Universitat Jaume I,
Campus Riu Sec, 12080 Castellón de la Plana, Spain
{castellp, jromero, chover}@uji.es
http://graficos.uji.es

Abstract. Nowadays computer graphics hardware presents a series of characteristics, such as AGP memory, vertex cache, etc., that can be used for real-time rendering. The aim of this paper is to conduct a comparative study of different techniques that are shown in the OpenGL graphics standard together with hardware features that enable the visualization of the geometry of complex objects to be accelerated. These techniques are applied to the multiresolution modeling, which requires techniques that can be implemented with dynamic geometry.

1 Introduction

Computer graphics systems present a pipeline architecture [12] in which 3D data about the scene that is to be displayed cross at different stages. Depending on the type of application or the nature of the data, it is possible that some of these stages become a bottleneck, thus drastically reducing the overall performance of graphics devices. It is therefore essential to locate these bottlenecks so that the graphics hardware can operate properly.

Figure 1 shows, in diagram form, the different stages of a graphics pipeline architecture that OpenGL presents [13] [7].

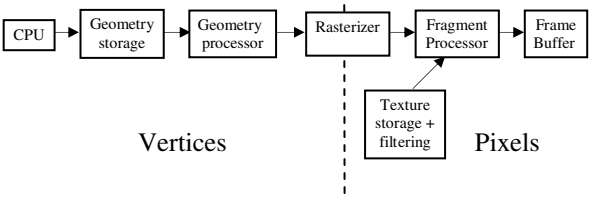


Fig. 1. Simplified OpenGL graphics pipeline

In this architecture, the main bottlenecks that can be distinguished are as follows:

- CPU. There are problems with the application or the driver.
- AGP/PCI Express. There are too many data for the bus.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 240 – 247, 2005.
© Springer-Verlag Berlin Heidelberg 2005

- Geometry. There are too many vertices or too much computation per vertex.
- Rasterization. There are also too many large-sized triangles.
- Texture. Textures are too big, texture cache is underutilized and the filters are expensive.
- Fragment Shader. There are too many fragments or too much computation per fragment.
- Frame buffer. There is too much read/write to the frame buffer.

The first step is to find the bottleneck and later eliminate it, if possible. If this cannot be done, we then have to try to balance the pipeline.

Traditionally the problem of representing large objects is solved by using multiresolution models [4] [11]. In the multiresolution modeling an object is represented by different approaches, each from a different level of detail (LOD), which is going to necessarily generate bottlenecks in the processing of vertices. There are two types of multiresolution models: continuous, in which geometry changes in runtime as a result of the variation of the level of detail, and discrete, where a few levels are precalculated and one level or another are shown according to the visualization requirements of the scene. The first one requires techniques for dynamic geometry and the second can benefit from techniques for the visualization of static geometry.

The specific problems within geometry are usually the following:

- Wrong transfer due to the use of non-native data types, inadequate sizes or calls to the graphics libraries using less efficient primitives.
- Geometry transformation that gives rise to bottlenecks due to the number of vertices and to the calculations based on them, or because of an inefficient use of the vertex cache.

Within this framework we intend to perform an in-depth analysis of the problems related to geometry, the solutions currently available and their possible application to dynamic geometry.

In section 2, several solutions offered by OpenGL are presented, as well as general solutions related to the drawing primitive, such as the use of extensions that make use of video memory and the vertex cache included in modern graphics processors (GPU). Section 3 describes the solutions applied to several test cases in this paper, and in section 4 the results that were obtained are reported. Finally, conclusions and future work are presented in section 5.

2 Solutions Overview

Different solutions are available with which to resolve the bottleneck problems in the geometry subsystem. They can be classified as specific solutions, which are directly linked to the graphics library that is used, and general solutions, which are more related to the type of primitive employed in the visualization of 3D objects or to the particular characteristics of modern GPUs.

2.1 Specific Solutions

Among the specific solutions, the following can be considered in OpenGL [7]:

- **Immediate mode.** This consists in drawing geometry by means of glBegin()...glEnd() operations. This is the primitive with lowest performance and the least recommended, as is shown below.
- **Display list.** A display list is simply a command group and OpenGL arguments that have been saved in a suitable fashion for later execution by the hardware. Its fundamental characteristic is that it cannot be modified. Obviously the main limitation of this solution is that in principle it is not adapted to dynamic geometry because any change in geometry implies the creation of a new display list. However, it is very well adapted to the representation of static geometry or discrete multiresolution models.
- **Vertex array.** This consists of storing the associated vertex data in special arrays, so that these arrays can be used to specify several geometries with the execution of only one command. The array of vertices can only contain the vertices of the geometry and a list of indexes is needed to generate the primitives based on that array. This scheme is suitable to exploit the characteristics of dynamic geometry because the array of vertices is generated only once and it is then possible to modify the list of indices at runtime.
- **OpenGL extensions.** Extensions enable new characteristics and capabilities to be added to OpenGL. In general, what the extensions related to geometry acceleration attempt to do is to load the data concerning the vertices into the local memory of the graphics card and try to avoid using the main memory.

  Of the extensions that have been developed by manufacturers, the following are perhaps the most noteworthy:

  - ATI_vertex_array_object (8/2002)
  - NV_vertex_array_range (9/2001)

  An extension that has recently appeared with the same intention as the previous ones and which has been officially approved by OpenGL is ARB_vertex_buffer_object (2/2003). This is the extension that was used in this study; its use allowed the behavior of OpenGL vertex arrays to be modified by letting them reside in the local memory of the graphics card.

2.2 General Solutions

Graphics processors have hardware characteristics that allow them to represent 3D objects suitably by means of triangles.

In order to avoid sending repeated vertices to the graphics system other structures such as triangle strips or triangle fans can also be employed. In a triangle strip, the first triangle is drawn and then only one vertex is added for each new triangle that is added to the strip. In a triangle fan, all the triangles share one vertex they have in

common. Consequently, the visualization of an object by means of triangle strips or fans achieves greater acceleration because less information is sent to the graphics system.

In this paper we have analyzed the following algorithms that allow us to obtain a strip-based representation:

- Stripe [3] (Stony Brook University)
- NvTriStrip [8] (NVIDIA Corporation)

Both Stripe and NvTripStrip allow strips to be generated, but only NvTriStrip takes advantage of the vertex cache. Current GPUs have a series of registers that allow us to store the last 16 or 24 vertices used. It is possible to try to provide the graphics system with information about vertices ordered in such a way as to produce the smallest number of page faults. Several studies have been carried out that try to take advantage of this characteristic. These include Chow [2], who presents a method for the compression of geometry; Hoppe [6], who develops an algorithm for the generation of strips while taking advantage of the vertex cache in a transparent way; and Bogomjakov and Gotsman [1], who present a method for the optimization of the vertex cache applied to the Progressive Meshes [5] multiresolution model.

The NvTriStrip library [8] allows both triangles and triangle strips to be generated, taking advantage of the GPU vertex cache and, in addition, it allows for the generation of one single triangle strip (by using degenerated triangles) or several triangle strips.

The Stripe utility [3] allows triangle strips to be generated from objects represented by means of the obj format. This utility always generates several strips, so it does not make use of degenerated triangles.

3 Description of the Comparison

The objective of this study was to measure the performance of all the techniques described above for 3 test meshes: cow (5804 triangles), sphere (30 624 triangles) and bunny (69 451 triangles). These measurements consisted in obtaining the frames per second (fps) rates, that is to say, the number of times per second the system is able to draw the scene.

These techniques were applied to models represented as triangle lists and triangle strips, and the extension that refers to local memory buffers and GPU vertex cache was also tested.

First, we assessed the use of techniques classified as specific solutions:

- Immediate mode
- Display list
- Vertex array
- Vertex buffer object

These techniques were applied to the general solutions. The following aspects were compared within the triangles model:

- Original model triangles (respecting the order in which they appear in the obj file)
- Optimized triangles obtained by NvTriStrip
- One strip with degenerated triangles obtained by NvTriStrip
- Several strips obtained by NvTriStrip
- Several strips obtained by Stripe

Finally, the effect exerted by the vertex cache was measured. It was tested by using NvTriStrip with remapping (vertices are rearranged in the same way as indexes). In particular, times with and without cache were measured using different specific solutions:

- NvTriStrip optimized triangles
- NvTriStrip strips (one strip with degenerated triangles and several strips)

Lastly, the results obtained using the different techniques were compared.

4 Results

The experiments were conducted using different PCs: Pentium IV 2600 MHz, Pentium Xeon 2800MHz and dual Pentium III with 1 Gb RAM and several GPUs: NVIDIA GeForce FX5900[1] 128 Mb, GeForce FX5700[2] 256 Mb and GeForce 6600[1] 256 Mb with two operating systems: Windows XP Professional SP1 and Linux SuSE 9.1. They involved measurement of the number of frames per second (fps) that the system reaches using each of the techniques. The implementation was performed in C++ using the OpenGL graphics library. Ms Visual C++ 6.0 SP5 was employed as a compiler for windows and gcc 3.3.3 for Linux.

The results obtained for the 3 test meshes are shown in Tables 1 to 3.

Table 1. Results (fps) obtained for Cow

Cow	GPU	V. cache	t / s	Kb	IM	DL	VA	VBO
Triangles: original model	*FX5700*	no	5804	136	397.60	1315.68	707.68	1050.95
	Ge6600				850.00	1319.00	1259,70	1268.00
	FX5900				816.19	1951.05	1421.58	1913.09
Triangles: NvTriStrip	*FX5700*	yes	5804	136	371.63	1288.71	384.62	1031.97
	Ge6600				796,20	1488,50	1420,60	1430,60
	FX5900				774.23	1958.04	1385.61	1888.11
One strip: NvTriStrip	*FX5700*	yes	1	102	543.90	1408.59	831.17	1239.76
	Ge6600				1084,90	1494,50	1436,60	1434,60
	FX5900				1089.91	2154.88	1860.20	2081.92
Several strips: NvTriStrip	*FX5700*	yes	551	98	427.72	1368.63	701.30	1105.89
	Ge6600				1015.00	1430,60	1441,60	1427,60
	FX5900				981.02	2010.99	1550.45	1674.33
Several strips: STRIPE	*FX5700*	no	101	98	622.38	1592.41	833.17	1259.74
	Ge6600				1009.00	1520,50	1440,60	1450,60
	FX5900				966.03	2350.65	1995.00	2283.72

[1] Tested under Windows.
[2] Tested under Linux.

Table 2. Results (fps) obtained for Sphere

Sphere	GPU	V. cache	t / s	Kb	IM	DL	VA	VBO
Triangles: original model	*FX5700*	no	30 624	717	86.14	584.83	119.64	328.02
	Ge6600				216,10	834,33	758,20	828,17
	FX5900				205.38	1121.88	397.60	1143.86
Triangles: NvTriStrip	*FX5700*	yes	30 624	717	79.60	748.50	118.53	296.70
	Ge6600				212,40	861,14	810,20	849,15
	FX5900				204.80	1239.76	394.61	1197.80
One strip: NvTriStrip	*FX5700*	yes	1	512	132.60	830.17	178.64	590.82
	Ge6600				394,20	860,14	812,20	847,15
	FX5900				384.23	1287.61	772.23	1221.78
Several strips: NvTriStrip	*FX5700*	yes	1763	496	111.00	780.22	176.47	520.96
	Ge6600				324,40	773,23	721,20	764,24
	FX5900				335.33	1271.73	645.35	986.01
Several strips: STRIPE	*FX5700*	no	172	481	165.41	785.21	213.14	601.40
	Ge6600				336,30	831,17	863,20	765,23
	FX5900				296.11	1149.85	870.13	1049.95

Table 3. Results (fps) obtained for Bunny

Bunny	GPU	V. cache	t / s	Kb	IM	DL	VA	VBO
Triangles: original model	*FX5700*	no	69 451	1630	31.40	218.13	29.01	113.10
	Ge6600				94,53	256,50	112,10	172,70
	FX5900				75.40	345.96	187.62	247.75
Triangles: NvTriStrip	*FX5700*	yes	69 451	1630	36.31	420.00	32.35	162.67
	Ge6600				99,50	512.00	337,70	506.00
	FX5900				93.91	763.24	185.25	741.26
One strip: NvTriStrip	*FX5700*	yes	1	1218	56.10	527.42	55.56	256.23
	Ge6600				171,50	501,50	366,90	495,50
	FX5900				160.20	778.22	360.64	758.48
Several strips: NvTriStrip	*FX5700*	yes	6194	1168	44.12	485.03	69.86	217.13
	Ge6600				121.00	423,70	274,50	455,10
	FX5900				130.61	768.23	270.73	438.12
Several strips: STRIPE	*FX5700*	no	917	1176	58.88	562.87	73.05	286.71
	Ge6600				119,80	499.00	401,80	431,70
	FX5900				105.89	715.57	335.33	614.39

Table 4. Techniques for geometry acceleration in LodStrips, results mesured in milliseconds

Object	Test	Vertices	Strips	Mb	IM	VA	VBO
Cow	*Linear*	2 904	136	0.183	3.74	2.56	1.01
	Exponential				4.60	1.29	1.06
Bunny	*Linear*	34 834	1 229	2.111	19.06	5.85	5.19
	Exponential				24.74	7.47	6.65
Horse	*Linear*	58 485	1 964	3.098	26.38	8.54	7.57
	Exponential				35.51	13.17	9.85

Finally, we have also experimented with the use of acceleration techniques in a continuous multiresolution model, based on strips, called LodStrips [9]. We have applied the linear test and the exponential test [10] for the following test meshes:

Cow, Bunny and Horse. We have considered the total times (extraction + rendering) in milliseconds to measure the performance.

As shown in Table 4, if we consider the object of highest complexity (Horse), it can be seen that the vertex array technique improves performance by up to 300% with regard to the immediate mode. The use of VBO provides an increase in performance about 20% with regard to the vertex array technique.

5 Conclusions and Future Work

Of the different visualization techniques used by OpenGL, the worst technique is the immediate mode. Vertex array leads to a gain of at least 300%, although the technique that achieves the best performance is the display list, which offers a gain of 800% with regard to the immediate mode. The use of VBO remarkably improves the performance of the vertex array by up to 700% with regard to the immediate mode, although it is lower than display lists. The use of the vertex cache significantly improves VBO performance in complex 3D models, and almost reaches the level of display lists. These proportions remain the same regardless of the drawing primitive used, that is, whether they are strips or triangles.

In relation to the drawing primitive, the use of strips instead of triangles improves performance by at least 100% if optimized triangles are used. If we employ original model triangles, however, this can reach 600% when the drawing primitive is a display list. On the other hand, the memory requirements of strips are also about 30% lower than in the case of triangles.

Finally, on comparing NvTriStrip strip generation with that of Stripe, it can be observed that the benefits deriving from the best NvTriStrip case (one single strip with degenerated triangles) and Stripe are similar, although NvTriStrip is significantly improved in VBO. Stripe generates fewer strips than NvTriStrip. For this reason, it achieves higher performance than NvTriStrip with several strips.

Therefore, it can be concluded that using strips as a drawing primitive and display lists for static geometry is the best way to obtain the highest performance. In a case involving dynamic geometry, VBO may be employed since its use in conjunction with the vertex cache offers an increase in performance of 775%, compared with the immediate mode, and this is close to the value achieved by display lists (an increase of 800% with regard to the immediate mode).

Lastly, it can also be concluded that the application of the described techniques to multiresolution modeling improves performance. However, these techniques impose

Table 5. Summary of performance in relation to the immediate mode

Primitive	DL	VA	VBO	VBO with NvTriStrip
Original triangles	300%	150%	225%	-
Optimized triangles	775%	200%	650%	700%
Triangle strips	800%	300%	700%	775%

special requirements when working with data structures. This is notable at the time of defining a new multiresolution model, and we must consider these restrictions if we want to reach optimal performance. LodStrips is a multiresolution model in which these techniques are applied easily.

An extension to this paper would be the application of the techniques described above to different multiresolution models in order to increase their performance.

Acknowledgments

We are sincerely grateful to Cem Cebenoyan of NVIDIA Corporation for his helpful feedback concerning the NvTriStrip library.

This work was partly financed by projects GameTools IST-2-00463 from European Comission, Mater TIN2004-07451-C03-03 from the Spanish Ministry of Science and Technology, VisualCAD FIT-3501101-2004-15 from the Spanish Ministry of Industry, Tourism and Trade, and Juegos P11B2004-22 (Fundació Caixa Castelló - Bancaixa).

References

1. A Bogomjakov, C Gotsman. “Universal Rendering Sequences for Transparent Vertex Caching of Progresive Meshes”, Computer Graphics Forum, (21:2) (2002) pp. 137-148.
2. M M Chow. “Optimized Geometry Compression for Real-time Rendering”, Proceedings of the IEEE Visualization ’97 (1997), pp. 347–354.4. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996).
3. F Evans, S Skiena and A Varshney. Optimising Triangle Strips for Fast Rendering, IEEE Visualization ’96, pp. 319-326, 1996. http://www.cs.sunysb.edu/~stripe
4. M Garland. Multiresolution modeling: survey & future opportunities. State o the Art Reports of EURO-GRAPHICS ’99, 1999.
5. H Hoppe. “Progresive Meshes”, Proc. of SIGGRAPH ’96 (1996) 99-108.
6. H Hoppe. “Optimization of Mesh Locality for Transparent Vertex Caching”, *ACM SIGGRAPH 1999*, pp. 269-276.
7. R Kempf and C Frazier. OpenGL reference manual. Reading, Addison-Wesley Developers Press, 1997.
8. NvTriStrip Library, NVIDIA Corporation (2002). Available on the Internet at the following URL : http://developer.nvidia.com/object/nvtristrip_library.html
9. F Ramos, M Chover, LodStrips, Lecture notes in Computer Science, Proc. of Computational Science ICCS 2004, Springer, ISBN/ISSN 3-540-22129-8, Krakow (Poland), vol. 3039, pp. 107-114, June, 2004.
10. J Ribelles, M Chover, A López and J Huerta, A First Step to Evaluate and Compare Multiresolution Models, Short Papers and Demos of EUROGRAPHICS’99 (1999) 230-232.
11. J Ribelles, A López, O Belmonte, I Remolar, M Chover. Multiresolution modeling of arbitrary polygonal surfaces: a characterization. Computers & Graphics. Elsevier Science. 2002.
12. J Spitzer, “OpenGL Performance Tuning”, NVIDIA Corporation, GameDevelopers Conference 2003.
13. M Woo, J Neider, T Davis. OpenGL programming guide. Reading, MA: Addison-Wesley Developers Press, 1997.

Building Chinese Ancient Architectures in Seconds

Hua Liu, Qing Wang, Wei Hua , Dong Zhou, and Hujun Bao

State Key Lab of CAD&CG, Zhejiang University
{sun_day, qwang, huawei, zddong, bao}@cad.zju.edu.cn

Abstract. Chinese Ancient Architecture (CAA) has a remarkable feature that its construction follows a set of rigorous constraints. Based on this property, we present a procedural approach for fast CAA modeling. Knowledge of the architecture construction is formalized as a set of rules and the primary features of buildings are parameterized to control the modeling process. By this approach, it is efficient and convenient to generate regular architectures or a class of architectures with a similar style. A system is implemented based on this idea. More than one hundred different buildings have been generated by a rule set comprising of about 120 rules.

1 Introduction

Chinese Ancient Architecture (CAA) is well known for its gorgeous appearance and contraption. However, it is really tedious to model it with mainstream modeling software, since specific knowledge of CAA construction is required. More over, the construction of CAA is very complex, as illustrated in Figure1.

By exploring the intrinsic discipline of CAA, we present a procedural approach for fast modeling, which significantly simplifies the traditional modeling process. The knowledge of the architecture construction is formalized in a set of rules. The building process consists of iteratively interpretation of rules and user-specified style parameters. Using our system, even novices with little specific knowledge can also generate different kinds of CAA by simply setting a few control parameters. Figure 7 shows a typical scene which is generated in less than two seconds.

1.1 The Feature of Chinese Ancient Architectures

Chinese ancient architecture has a remarkable feature that both the architecture itself and its layout follow a set of rigorous construction constraints. Although different architectures with different grades, races, regions or eras, have their own features, from the view of structure, they follow a whole set of construction constraints which were illuminated in the book *Yinzaofashi*[13, 14].

As the fact that Chinese ancient architecture follows a set of rigorous construction constraints, we could use a set of modeling rules to formalize the construction constraints. Given some initial constraints, CAA models can be generated rapidly by procedural modeling approach based on these rules. Furthermore, using this modeling method, modeler needs little architecture knowledge.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 248 – 255, 2005.
© Springer-Verlag Berlin Heidelberg 2005

1.2 Related Work

Our approach for modeling architectures is one of the procedural modeling techniques. L-System or parameter L-System, which is mainly used in plant modeling [1, 2, 3], is a predominant one among them. In [4], an extended L-System was used to generate streets of a city. Population density, height map and water map of a city were used as the input parameters that influenced the generation of the streets.

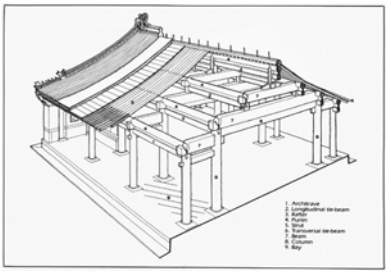


Fig. 1. The picture of an Chinese Ancient Architectures

Shape grammar or *parameterized shape grammar* [5], was a powerful analysis and design tool for architectures. Using this tool, modeling rules can be abstracted from the existing buildings [6, 7] and new architectures can be designed by a set of modeling rules [8, 9]. However, the rules written by the shape grammar are directly based on 2D/3D shapes. It is difficult to identify and represent the shape rules by computer. Selecting and deducing rules only can be executed by architects. As a consequence, it is not suitable for automatic modeling of architectures.

Split grammar [10] was a subset of the shape grammar. It describes the rules of shape splitting, which is used to produce a class of models. *Control grammar* was introduced to distribute the attributes of the modeling rules and selecting the next executed rule by matching attributes. The modeling process starts from a bounded simple shape, and then splits it recursively to form a complex shape with many details progressively.

Another technique in architecture modeling is Image-Based Modeling (IBM). The realistic models can be recovered from photographs [11, 12]. However, these methods need human interactions to identify the pairs of corresponding points. It decreases the modeling efficiency in practice. This method can be used to rebuild architectures, but is not designed to create new models without photographic input data.

1.3 Our Approach

The idea of our approach is from the shape grammar. As it is hard to describe the shape grammar by computer completely, we try to define a subset of it, which is named constructive grammar.

Constructive grammar (CG) is different from the split grammar (SG) in some aspects. The main difference between them is the modeling process. The SG starts from a bounded simple shape, and then splits it recursively to form a complex shape with many details, while the CG starts from an unbounded space and produces an architecture model by combining some kinds of basic shapes progressively. The process of the CG is like building toy bricks which we played in our childhood. Additionally, the SG emphasizes designing surface detail of architecture, while the CG focuses on building the whole structure of the architecture, including its exterior and interior geometry information. As illustrated in Figure 5, the result models built by our approach include not only the exterior geometry information, but also the interior structure information.

2 Constructive Grammar

Constructive grammar introduced in this paper is a specialized form of the shape grammar [4], which is designed to procedurally model buildings. Its utility stems from the fact that the restrictions of architectures have been carefully chosen so as to make automatic rule selection possible. The atomic object manipulated by the grammar is the component.

2.1 Constructive Grammar

- **Definition 1:** A *primitive* is a parameterized basic shape.
  Buildings always consist of a set of basic shapes, such as cubes, cylinders, prisms and so on. These basic shapes are called primitives. The geometry of a primitive is parameterized, which is used to represent the geometry of the component.
- **Definition 2:** A *component* is a basic building block of an architecture model. It is represented by a quad-tuple $< n, p, A, s >$, where n is the component name, which indicates its semantic meaning; p is a primitive; A is a finite set of attributes and s describes the state of component.

  The name n distinguishes the components from each other according to their functions. The primitive p indicates the geometry of the component. The state s is an enumerative variable that records the information of the component during a modeling process, which helps to control the modeling process.
- **Definition 3:** A *component list* is a set of components with a sequence, where the sequence indicates the building order of these components.

  A simplest component list is composed of only one component.
- **Definition 4:** *Constructive Grammar* is defined by $< M, F, R, S, \$ >$,where

  1. M is a finite set of components;
  2. F is a finite set of restrictions;
  3. R is a finite set of rules in the form $\alpha \rightarrow \beta$, where α is a component and β is a component list;
  4. S is a finite set of occupied space of a component list;
  5. $\$$ is an initial space in S;

The vocabulary of the constructive grammar is the set $V = \{t(m) \mid m \in M, t \in T\}$, where T is the set of transform e.g. rotation, translation and reflection. The restrictions in F are abstracted from a class of buildings and formalized as some Boolean functions.

R is a set of rules. Here, the rules are formalized as the productions in the extension L-System. All the rules are classified into two classes: *combining rules* and *conversion rules*, based on their functions. Combining rules are used to add new components and keep the existing ones, while conversion rules are used to replace the existing components with new ones. As illustrated in Figure 2, after a conversion rule C3 is used, a PILLARBOX is converted into a PILLAR. However, a combining rule B2 is used to create a PILLARBOX on the existing BASEBOX.

Each element s in S is the occupied space of a component list. It is a specific component, which is used to locate the corresponding component list in 3D space. In our approach, the bounding box of a component list is used to represent s and the geometry of s is a cube primitive. $ is specified by user, determining the initial space which an architecture model occupied.

2.2 A Simple Example

An example of modeling a pillar, which is illustrated in Figure 2, is given in this section and the corresponding rules are shown in appendix A.

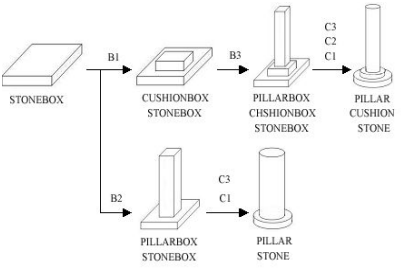


Fig. 2. The modeling process of a pillar. The strings above the arrow lines are the rule numbers of the executed rules

The symbol $ determines the initial size and position of the model. The tokens BASEBOX, CUSHIONBOX and PILLARBOX, which belong to the set S, represent the space occupied by the corresponding component BASE, CUSHION and PILLAR. Rules B1, B2 and B3 are the combining rules to build the connectivity of components. Meanwhile, the rules C1, C2 and C3 are conversion rules to determine the actual form of the model.

The restrictions STATE == BEGIN and STATE == FINISHED both use the state information STATE of components as their parameters. The STATE of component BASEBOX indicates whether the BASEBOX has a PILLARBOX or CUSHIONBOX on it. The rules B1 and B2 will be selected and executed only when the STATE of component BASEBOX equals to BEGIN, which indicates that the BASEBOX has no PILLAEBOX or CUSHIONBOX on it. In this example, we only use the modeling state of components to indicate the restrictions. In other situations, position and direction of the components are also applied in restrictions.

The expression $1.0-x$ in rule B1 and the value 0.4 in rule B1 are the probability of the rule for the probability comparison. x is a variable randomized between 0.0 and 1.0 during the modeling process.

3 A Modeling System for Chinese Ancient Architectures

In this section, we will show you how to build Chinese Ancient Architectures with the constructive grammar.

3.1 Abstract Modeling Rules

3.1.1 Local Pattern

The styles of CAAs are mainly showed in three aspects: Wall、Pillar and Roof. Each of them has its own pattern, which represents the basic structure variation in CAAs.

- Pillar pattern: To decide the building's pillar layout. As illustrated in Figure 3, it has four different patterns;
- Wall pattern: To decide the relations between walls and pillars of a building. As showed in Figure 4, three patterns are included;
- Roof pattern: To decide the building's roof style. CAAs have mainly six roof patterns and we adopted three of them in our system.

In our system, corresponding to each pattern, a subset of rules are created to describe it.

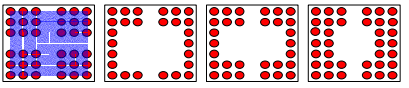


Fig. 3. It shows four pillar patterns, in which dot means pillar and box means the building body's room. In (a), the rectangle stands for ***KAIJIAN***. In this example the *AKR* is 6 and *AKC* is also 6

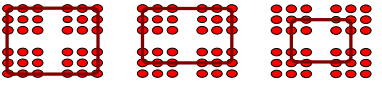


Fig. 4. It shows three wall patterns, in which the pillar pattern is as same as the one shown in figure 3(a). Dot means pillar, while line means wall

3.1.2 KAIJIAN

KAIJIAN is a professional glossary in CAAs. It can be regarded as the unit space in a building. In Figure 3, the area of the small rectangles are KAIJIANs. *The Amount of KAIJIAN in Row*(AKR) and *the Amount of KAIJIAN in Column* (AKC) decide the scale of the building.

3.1.3 The Generation of Rules

In our system, we took AKR, AKC and three local patterns as control parameters. Each parameter could choose three to five values. Using these control parameters, more than 100 different building models could be generated.

The constructive rules were created according to [13, 14]. The architecture elements such as door, pillar, baluster and so on, were all treated as components in the constructive grammar. And all the parameters of them were described as attributes. The bounding box of the component was written in rules as the element in the set S. For generating layout of buildings, the arrangement of buildings was also considered.

Totally, about 10 primitives, 30 components and 30 attributes were defined to create the rule set. It consists of about 120 modeling rules, in which 110 rules are for modeling buildings and the rest are for creating buildings' layout.

3.2 Implementation

The interpreting process of our system is similar to the extension L-system. The rules are interpreted by a string parser and a visualization parser. All the rules are managed

by a hash-table, which accelerates rules selection. In the hash-table, the name of the component is used as the key to search rules. The rules with the same key are further organized to a list sorted by the rule's probability. The symmetric character of Chinese Ancient Architecture is exploited to accelerate the modeling process.

Textures are assigned to the corresponding components based on their semantics meanings. One look-up table is used to store the relations between textures and the corresponding components. The name of the component acts as the search key for finding the correct texture in texture mapping. In texture coordinates computing, the relation between the size of texture and the dimension of the corresponding component is considered.

3.3 Experimental Results

Our system was tested on a Windows platform with AMD AthlonXP 1800+ CPU,ATI Radeon 9100 graphics card and 512MB memory.

There were five control parameters in our test. By changing the value or the combination of these control parameters, more than one hundred kinds of constructions could be obtained. Additionally, architecture models of different styles could be generated by applying different textures. The Figures[5,6,7] show some models generated by our system (All the models were rendered in 3D studio MAX) .

Fig. 5. Bamboo house. The control parameters of the model are 7AKR & 5AKC and the generating time is 0.381s. Left: the outside of the bamboo house; Right: the inside of the bamboo house

Fig. 6. Left: Civilian house. The control parameters of the model are 5AKR&5AKC and the generating time is 0.24s ; Right: Palace. The control parameters of the model are 9AKR & 5AKC and the generating time of it is 0.972s

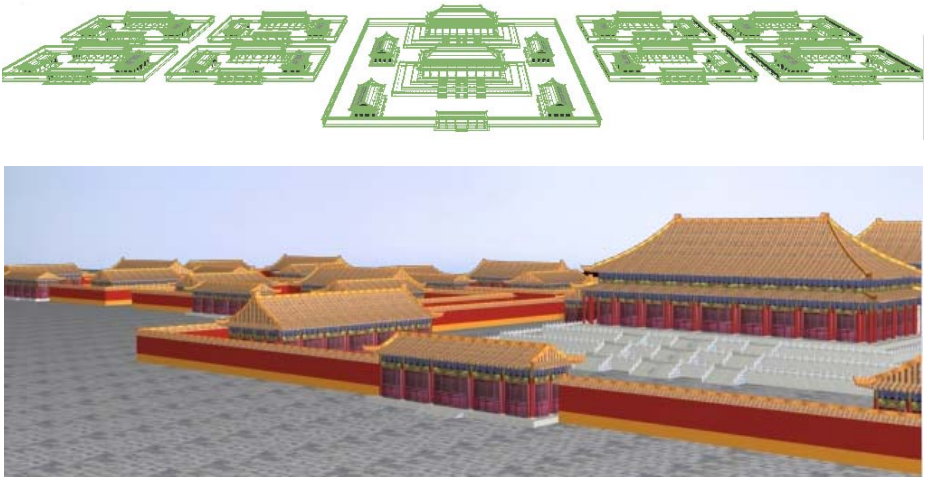


Fig. 7. The image shows a corner of the Forbidden City built by our system. The generating time of it is 1.932s

4 Conclusion and Future Work

In this paper, a rapid procedural modeling approach to build CAAs is represented, which is based on the constructive grammar. This approach is suitable for modeling regular architectures or a class of architectures with some common features, as well as for generating layouts which consist of buildings with a similar style. More than One hundred buildings were generated by about 110 modeling rules.

The most important step of our approach is to abstract the semantic rules from architectures. In order to do that, we must analyze the features of buildings and formalize them into semantic rules. Sometimes, it is not intuitive enough to acquire the rules. In the future, we will explore a method to abstract rules conveniently or even automatically by machine learning techniques. Additionally, a texture synthesis approach driven by semantic rules is considered to be integrated into the current system in the future.

Acknowledgements

We acknowledge Prof. Andrew I-kang Li, in the Department of Architecture of the Chinese University of Hongkang, who has offered us many references about architectures. We also acknowledge the contributions from Mr. Huang xiang, in the Architecture Department of Zhejiang University. This work is supported in part by National 973 Program of China (2002CB312102), National Natural Science Foundation of China (Grant No. 60203014, 60021201).

References

1. Przemyslaw.P, James.M and Měch.R.: Synthetic topiary. In Proceedings of ACM SIGGRAPH 1994, ACM Press, A.Glassner, Ed, pp. 351-358.
2. Przemyslaw.P, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane: The use of positional information in the modeling of plants. In Proceedings of ACM SIGGRAPH 2001, ACM Press, pp. 289-300.
3. P. Prusinkiewicz and A. Lindenmayer: The algorithmic beauty of plants. Springer-Verlag, NewYork,Inc. 1990. New York,USA.
4. Yoav I.H.Parish and Pascal Müller: Procedural modeling of cities. In Proceedings of ACM SIGGRAPH 2001, ACM Press, E.Fiume,Ed, pp. 301-308.
5. G.Stiny: Introduction to shape and shape grammars. Environment and Planning B 7 (1980), Pion Ltd, London, England. pp. 349- 351.
6. G.Stiny: Ice-ray: a note on Chinese lattice designs. Environment and Planning B 4 (1977), Pion Ltd, London, England. pp.89-98.
7. G.Stiny and W. J. Mitchell: The Palladian grammar. Environment and Planning B 5 (1978), Pion Ltd, London, England. pp.5-18.
8. T.W.Knight: Designing with Grammars. In CAAD futures'91 Ed G. N. Schmitt, Computer-Aided Architectural Design (Verlag Viewag, Weisbaden, 1992), pp.33-48.
9. T.W.Knight: Color grammars: designing with lines and colors. Environment and Planning B: Planning and Design 16 (1989), Pion Ltd, London, England. pp.417-449.
10. Peter Wonka, Michael Wimmer, Francois Sillion, and William Ribarsky: Instant Architecture. ACM Transactions on Graphics, ACM Press, volume 22. number 3. pp. 669-677. July 2003
11. Paul E. Debevec, Camillo J. Taylor and Jitendra Malik: Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In Proceedings of ACM SIGGRAPH 96, ACM Press, H. Rushmeier, Ed., pp.11-20.
12. Paul Debevec, Yizhou Yu and George Borshukov: Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. Proceedings of 9th Eurographics Workshop on Rendering, Vienna, Austria, June 1998, pp.105-116. UC Berkeley Technical Report, UCB//CSD-98-1003
13. Liang Sicheng: Yingzao fashi zhushi [The annotated Yingzao fashi]. Beijing:Zhongguo jianzhu gongye,1983.
14. Liang Sicheng: Liang Sicheng quan ji. Chapter 4,6,7, Beijing: Zhongguo jianzhu gongye,2001.

Accelerated 2D Image Processing on GPUs

Bryson R. Payne[1], Saeid O. Belkasim[2], G. Scott Owen[2],
Michael C. Weeks[2], and Ying Zhu[2]

[1] Georgia College & State University, Department of ISCM, Milledgeville, GA 31061
bryson.payne@gcsu.edu
[2] Georgia State University, Department of Computer Science, Atlanta, GA 30303
{sbelkasim, sowen, mweeks, yzhu}@cs.gsu.edu

Abstract. Graphics processing units (GPUs) in recent years have evolved to become powerful, programmable vector processing units. Furthermore, the maximum processing power of current generation GPUs is roughly four times that of current generation CPUs (central processing units), and that power is doubling approximately every nine months, about twice the rate of Moore's law. This research examines the GPU's advantage at performing convolution-based image processing tasks compared to the CPU. Straight-forward 2D convolutions show up to a 130:1 speedup on the GPU over the CPU, with an average speedup in our tests of 59:1. Over convolutions performed with the highly optimized FFTW routines on the CPU, the GPU showed an average speedup of 18:1 for filter kernel sizes from 3x3 to 29x29.

1 Introduction

Programmable graphics processing units (GPUs) are commonly included as hardware components in new computer workstations. Furthermore, the current generation of GPU is technically superior in terms of GFLOPs of processing power [7] to that of current CPUs [2], but that power is rarely used to its full capabilities. Numerous advances have been made recently in applying the GPU to parallel matrix processing tasks, such as the Fast Fourier Transform [6]. This research applies the vector processing power of the GPU to 2D image processing convolution filters.

Section 2 briefly covers recent GPU advances, and discusses why the GPU is well suited for digital image processing applications. Section 3 explains how convolution on images can be achieved with a GPU, while section 4 compares the frames-per-second processing rates of an NVIDIA GeForce FX6800 to a dual 2.8 GHz Intel Pentium 4. Section 5 notes our conclusion of an average 18:1 speedup of the GPU over a CPU running optimized code on convolution filter mask sizes 29x29 and smaller, and a speedup of 59:1 over a CPU without optimized code.

2 Background

The first programmable consumer GPUs became available less than four years ago. The first generation of programmable GPUs was not well-suited to general-purpose

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 256 – 264, 2005.
© Springer-Verlag Berlin Heidelberg 2005

computation for several reasons. First, they allowed access to only selected portions of the graphics pipeline and had no easily accessible off-screen rendering capabilities. Second, they had to be programmed in GPU-specific assembly code, with no standardization across manufacturers. Third, their limited accuracy of 8 bits-per-pixel combined with their slower clock speeds and memory accesses, as well as smaller memory sizes, compared to CPUs made them unattractive to general-purpose computing researchers who needed fast 32-bit floating point operations as a minimum point of entry.

2.1 Computation on GPUs

By late 2002, a C-like programming language for the GPU, named Cg [5], had been developed for cross-platform GPU programming. Cg contained constructs for looping and conditional branching, required for most general-purpose computing, but GPU hardware took another two years to catch up to the capabilities provided for in the Cg language. Only in the NVIDIA GeForce FX6800 series GPU, released in late 2004, was it first possible to take advantage of true conditional branching on the GPU, as well as handle loops or programs that consisted of more than 1024 total instructions per pixel [7]. These advances enabled the research in this paper, and the previous work [8] upon which it is based.

Researchers at Stanford developed Brook for GPUs [1], a stream processing language abstraction for GPU programming. Unfortunately, stream processing is not especially well-suited to the limitations of current GPU hardware. Even the simple merge sort, a typical example of stream processing computation, is not possible in a single pass on current GPU hardware due to the inability to both read and write the same texture at the same time. This factor, along with others, caused Brook to show only a limited speedup on a number of test applications across platforms and manufacturers.

2.2 Digital Image Processing

Digital image processing (DIP) appears to be especially well-suited to current GPU hardware and APIs (application programming interfaces), due to the graphical nature of the GPU's processing power. Digital image processing is a field of computing that involves manipulating images that have been converted into digital form. Typical examples of DIP operations include smoothing or blurring an image, sharpening an image, edge detection, noise removal, and various effects such as embossing or beveling.

Figure 1 shows an original image with three types of filters applied. In addition to single 2D image filtering applications like those shown in Figure 1, DIP can be applied to video on a frame-by-frame basis, such as for motion detection. They can also be applied to 3D scenes generated on the GPU, such as when a flight simulator uses blurring to make a scene more realistic.

Many image processing operations are convolution-based, which means that every pixel of an image is processed by multiplying the elements of a smaller matrix, called a mask, with the values of the surrounding pixels, known as a neighborhood. For example, in a 3x3 averaging filter, the values of the nine pixels that comprise a pixel

and its eight surrounding neighbors within a one-pixel boundary are summed together and the result is divided by nine for the output pixel's value. This type of filter smoothes or blurs rough edges in an image, like the first example given in Figure 1.

Fig. 1. Examples of image processing convolutions. The original image (left) after blurring, sharpening, and edge detection are performed, respectively *(Image courtesy of Patrick Flynn)*

Convolutions become costly as the size of the filter mask grows. While 3x3 filters are useful in many applications, filter sizes as large as 65x65 and beyond are used, as well. On a 2048x2048 image, a total of 65x65x2048x2048 (over 17 billion) array lookups, multiplications and additions are needed to perform the straightforward convolution operation. Caching and other concerns quickly drive up the cost of this computation. Fortunately, it is possible to optimize the convolution operation by making use of Fourier processing, which will be discussed in further detail in the next section.

3 Implementation

The first step in comparing the GPU to the CPU in performing digital image processing operations will be to perform simple averaging filter convolutions across images ranging in size from 64x64 to 2048x2048. This size range includes the small image sizes common in texture maps developed for 3D games as well as the large image sizes that include full-screen views on the best available current display technology and consumer-level high-resolution digital camera images. In previous work [8], 3x3 filters were shown to have a significant speed advantage on the GPU over the naïve CPU implementation. Two new items remained to be addressed: how the GPU's advantage scales for larger filter sizes, and how the GPU compares to FFT-optimized CPU code.

3.1 DIP on GPU

Using Cg, programming a convolution filter on the GPU is a very straightforward process. Figure 2 shows a section of code for a simple 3x3 averaging filter in Cg. A texture lookup function (`texRECT`) makes it easy to find neighboring pixel values relative to the output pixel's coordinates (`texCoord`) by adding a vector (`float2`) to describe the relative position (up one pixel, left one pixel, etc.). Performing more advanced convolutions, such as edge detection, Gaussian smoothing filters, embossing, and more, only requires adding a multiplier to each texture lookup corresponding to the weight of the mask for that pixel's position.

The C version of a convolution, as shown in Figure 3, is more difficult for several reasons. First, the C version must treat each color component (red, green, and blue, or *RGB*) as separate computations, while the GPU computes all three components simultaneously as three-component vectors of type `float3`. Second, the C method computes array positions in a one-dimensional manner corresponding to linear addresses in main memory rather than the 2D relative indexing of Cg. Therefore, non-intuitive modular arithmetic is necessary to resolve pixel values. Furthermore, the Cg version does not depend upon the size of the image, due to the fact that the program is written to operate on a per-pixel basis. The C version must know the width and height of the image in order to perform the needed array index lookups.

```
float3 a[9];
a[0] = texRECT(image, texCoord + float2(-1, 1));
a[1] = texRECT(image, texCoord + float2(0 , 1));
…
a[7] = texRECT(image, texCoord + float2(0 ,-1));
a[8] = texRECT(image, texCoord + float2(1 ,-1));
color = (a[0]+a[1]+a[2]+a[3]+a[4]+a[5]+a[6]+a[7]+a[8])/9.0;
```

Fig. 2. A 3x3 averaging filter in Cg

```
int a1,a2,a3,a4,a5,a6,a7,a8,a9;
// wd and ht are the width and height dimensions of the RGB image im1
int m = wd*ht*3;
for (r=0; r<m;r++)
{
  a1=(r-(wd+1)*3+m)%m;
  a2=(r-(wd)*3+m)%m;
  …
  a8=(r+(wd)*3)%m;
  a9=(r+(wd+1)*3)%m;
  im2[r] = (im1[a1]+im1[a2]+im1[a3]+im1[a4]+im1[a5]+im1[a6]+im1[a7]+im1[a8]+
            im1[a9])/9;
}
```

Fig. 3. A simple subroutine in C for computing the 3x3 Averaging Filter

3.2 GPU Versus FFTW

Performing a fair comparison of the power of the GPU over the CPU in performing 2D convolutions involves more than the straightforward convolution. On the CPU, it is significantly faster in most cases to convert the image from the spatial domain to the frequency domain by means of the Fast Fourier Transform (FFT), then multiply the frequency representation of the image by the frequency domain version of the convolution filter, then take the inverse FFT, or IFFT, to acquire the filtered image.

$$F(u,v) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)[\cos(2\pi(ux+vy)) - i\sin(2\pi(ux+vy))] \quad (1)$$

The Discrete Fourier Transform (DFT) of an NxN image whose pixel values are stored in a matrix *f(x,y)* is described by the formula given in Equation 1 above. The Fourier Transform converts a 2D image in the spatial domain into a 2D matrix of frequency values. This frequency representation of the image is useful due to the fact that a spatial convolution of a filter mask of size MxM over an image of size NxN translates to a multiplication of the frequency domain representations of the two matrices. The FFT is simply a time-optimized approach for computing the DFT of an image.

A spatial 2D convolution of an image matrix *f* of dimension *NxN* and a filter mask *h* of size *MxM* is performed according to the formula given in Equation 2 below. Notice that every element of the image matrix is multiplied by every element of the mask matrix, for $O(N^2M^2)$ operations. For large filter mask sizes, this approaches $O(N^4)$ time. The FFT makes it possible to obtain the frequency domain representation of an image in $O(N^2 log^2 N)$. Assuming a pre-computed FFT of the filter mask, the element-wise multiplication of the image and mask matrices in the frequency domain takes only N^2 multiplications, followed by the IFFT, which involves the same computation time as the FFT, for a total of $O(N^2 log^2 N)$ time, which can be significantly faster than $O(N^4)$ for realistic values of N.

$$g(x,y) = \sum_{s=-w}^{w} \sum_{t=-w}^{w} h(s,t) f(x+s, y+t), \text{ where } w = \frac{M-1}{2} \tag{2}$$

In most image manipulation and processing packages, Fourier-domain processing is used to perform convolutions expressly because of the acceleration factor. For example, in a first trial run using the naïve convolution of a 15x15 mask on a 1024x1024 image took 6.5 seconds, while the FFT version took 0.54 seconds to perform the same convolution using the frequency-domain approach, a speedup of more than 12:1.

The steps necessary to convolve a filter over an image in the frequency domain are as follows. First, the mask is padded with zeros to match the size of the input image, and the FFT of the mask is computed. Because this only needs to be done once for each filter and size, this is usually precomputed and does not appear in the processing times given in the results. Second, the input image is converted to its frequency representation by performing the FFT. Note neither the mask nor the image should be shifted to the center, as is frequently done in Fourier analysis (if the filter mask is placed at the center of the zero-padded matrix, the input image must be shifted by multiplying every odd element by –1 before the FFT is performed). Third, an element-wise multiplication of the two matrices is performed. Fourth, the inverse FFT is performed on the resulting matrix. Fifth, the complex components of the IFFT are dropped, and the real values now contain the result of the image convolution. A highly optimized FFT package [3] will be used to test the speedup of this approach.

4 Results and Discussion

Straightforward algorithms for convolutions on the GPU and on the CPU were developed for comparison across a wide variety of matrix and image sizes (64x64 to 2048x2048). The GPU proved to be much faster than the CPU at straightforward convolutions like Gaussian smoothing, averaging filters, and edge detection for filter

masks from 3x3 to 29x29. Optimized CPU code was developed using the FFTW library, version 3.0.1 [3]. In the case of a 2D 3x3 convolution filter, the GPU (an NVIDIA GeForce FX6800) was up to 130 times faster at high resolutions (2048x2048) than the CPU (a dual 2.8 GHz Intel Pentium 4). The GPU was from four to 139 times faster than the FFTW-optimized version at performing a 3x3 convolution across images from 64x64 to 2048x2048. Figure 4 shows a logarithmic scale comparison of the GPU and FFTW versions over the naïve CPU version. Note that at 2048x2048 the CPU versions process less than one frame per second (fps), while the GPU processes over 60 fps, sufficient for real-time interactivity. Frame rate values are shown for the GPU and FFTW versions; naïve CPU numbers are given in Figure 5.

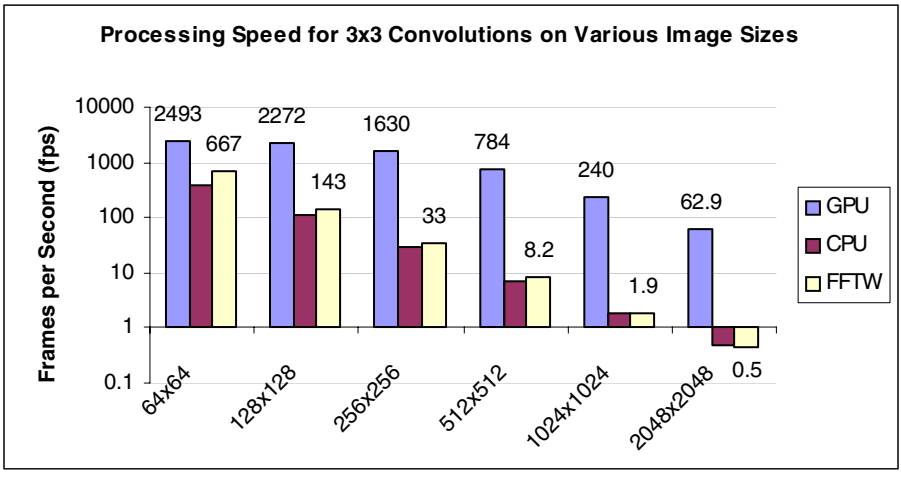


Fig. 4. Logarithmic scale speed comparison of convolving a 3x3 filter mask over images of varying sizes on GPU vs. naïve CPU convolution and FFTW-optimized Fourier processing

The GPU is especially well-suited to performing 2D convolutions and morphological masking and filtering operations. Furthermore, programming the GPU version of these algorithms is a straightforward process, allowing the developer to access pixel neighborhoods using a relative indexing paradigm rather than a complicated modular arithmetic scheme for referencing 2D array elements in main memory. The GPU provides the greatest speed advantage at smaller convolution kernel sizes in the 3x3 to 29x29 range, where the FFT provides the least advantage over the CPU. The GPU could therefore provide significant acceleration in applications where filter masks are small (edge detection, etc.) and where a compatible GPU is available.

Figure 5 shows the results of applying averaging filters of sizes ranging from 3x3 to 29x29 on images from 64x64 to 2048x2048. The table entries show the results for the GPU, followed by the naïve CPU convolution, in each cell. The bottom row shows the time for the FFTW-optimized filters. The FFTW version is given once for each image size due to the fact that the FFT approach to convolution does not vary in processing time by filter size, because each filter must be padded with zeros to match

the size of the input matrix before Fourier processing in order to multiply by the image matrix in the frequency domain. Therefore, it is just as fast to process an image with a 65x65 filter as it is to use a 3x3 filter when using Fourier domain acceleration.

Table 1. Timing results for the convolution of various filters (3x3 to 29x29) over various image sizes (64x64 to 2048x2048) on GPU/CPU (in frames per second)

Frame rates for applying convolution masks over various input image sizes (in fps)						
	64x64	128x128	256x256	512x512	1024x1024	2048x2048
3x3	2493 / 388	2272 / 111	1630 / 29.7	784 / 7.27	240 / 1.9	62.9 / 0.48
5x5	2293 / 299	1877 / 85.8	1019 / 22.3	358 / 5.56	96.7 / 1.4	23.6 / 0.35
7x7	2008 / 169	1433 / 45.6	634 / 11.5	193 / 2.93	50.1 / 0.73	12.3 / 0.18
9x9	1739 / 108	1099 / 28.2	422 / 7.21	120 / 1.79	30.3 / 0.45	7.2 / 0.11
15x15	1077 / 35.9	534 / 9.92	166 / 2.52	44.3 / 0.62	11 / 0.15	2.59 / 0.038
17x17	929 / 30.5	437 / 7.73	131 / 1.95	34.1 / 0.48	9.6 / 0.12	2.06 / 0.029
19x19	796 / 24.3	360 / 6.18	100 / 1.55	27.5 / 0.38	6.8 / 0.095	1.65 / 0.024
29x29	415 / 10	168 / 2.53	46.6 / 0.63	11.9 / 0.16	2.98 / 0.039	0.72 / 0.009
FFT	*667*	*143*	*33*	*8.2*	*1.9*	*0.45*

Figure 6 shows the speedup derived from dividing the GPU frame rates by the naïve and FFTW-optimized CPU frame rates. Note that the GPU is from 6 to 132 times faster than the straightforward CPU convolution, with an average speedup over these values of 59:1. The GPU is faster than even the FFTW implementation by an average of 18:1 for these filter sizes. At 29x29, however, the FFTW version begins to match the GPU in speed. Interestingly, this is also the largest 2D convolution filter we were able to implement in Cg. The code for a mask at this size must perform 29x29, or 841 texture lookups, multiplications, and additions per pixel, or around 2500 pixel operations, with 3 components per pixel (*RGB*). While the specifications for the 6800 series GPU state a maximum operation count of 65,536, we were unable to successfully implement filters any larger than 29x29.

Table 2. Speedup achieved in Figure 5. Figures are for GPU over naïve/FFTW-optimized CPU versions, rounded to the nearest whole number

Speedup of the GPU over the naïve CPU / FFTW convolutions						
	64x64	128x128	256x256	512x512	1024x1024	2048x2048
3x3	6 / 4	20 / 16	55 / 49	108 / 96	126 / 130	132 / 139
5x5	8 / 3	22 / 13	46 / 31	64 / 44	69 / 52	68 / 52
7x7	12 / 3	31 / 10	55 / 19	66 / 24	69 / 27	68 / 27
9x9	16 / 3	39 / 8	59 / 13	67 / 15	68 / 16	66 / 16
15x15	30 / 2	54 / 4	66 / 5	71 / 5	71 / 6	68 / 6
17x17	30 / 1	57 / 3	67 / 4	70 / 4	80 / 5	71 / 5
19x19	33 / 1	58 / 3	65 / 3	71 / 3	71 / 4	69 / 4
29x29	42 / 1	66 / 1	74 / 1	76 / 1	76 / 2	73 / 2

It was mentioned that it was assumed that the FFT of each filter had been pre-computed for timing purposes. If a new filter is used on each image, a processing penalty of 30% or more will result from adding another FFT to accommodate the new filters. Another important distinction between the convolution on the GPU and the FFT-based approach is that the FFT produces a cyclical convolution when used as described above. For visualization purposes, this is usually close enough to a linear convolution to produce acceptable results. If, however, a true linear convolution is required, the FFT routine must pad the image with zeros to increase the image size to twice its original height and width, causing a four-fold decrease in speed.

One other hidden consideration exists when performing the FFTW-optimized version of these operations. The first time the FFTW package is run on a machine for a given image size, the library creates an optimization strategy specific to the machine's architecture, memory, and other attributes. For small matrix sizes, this planning step took only a second or two, but for the 2048x2048 case, it took over 3 ½ minutes. Fortunately, the FFTW library provides a mechanism for saving and recovering the optimal plan so that this process need only be performed once, but the time consideration can be significant for the first use.

5 Conclusions and Future Work

In convolution-based operations from digital image processing (DIP), the GPU showed an average speedup of 59:1 over the straightforward CPU convolution and 18:1 over the FFTW-optimized approach for filter sizes from 3x3 to 29x29 on images from 64x64 to 2048x2048. Notably, the 60 fps speed of performing 3x3 convolutions on 2048x2048 images on the GPU could enable better and cheaper real-time high-resolution applications for scene reconstruction, navigation, and security purposes. Besides speed-up, GPUs have a programming advantage in Cg over CPUs programmed in C. This is due to the fact that pixels can be referenced in Cg in a relative fashion, such as one pixel to the right and down from the current location, unlike the more complicated 1D memory array lookup calculations needed in C.

Because 2D DIP is well-suited to GPU acceleration, a fruitful area for future research would be developing an image processing API or adding GPU acceleration to an existing image processing package. An API or multi-step compiler system like Brook [1] designed specifically for DIP operations could be a significant step toward general-purpose utilization of the GPU as a parallel vector processor.

References

1. Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., Hanrahan, P.: Brook for GPUs: Stream Computing on Graphics Hardware, *Proceedings of ACM SIGGRAPH 2004.* ACM Press, (2004) 777-786.
2. Dalvi, A. Intel targets 4 GHz barrier. *Techtree*, online news service. Available online at http://www.techtree.com/techtree/jsp/showstory.jsp?storyid=3970. (2003).
3. Frigo, M., Johnson, S.G.: FFTW: An Adaptive Software Architecture for the FFT, *IEEE ICASSP Proceedings (3)*. IEEE Press, (1998) 1700-1703.

4. Jain, A.K.: *Fundamentals of Digital Image Processing,* Prentice Hall, Englewood Cliffs, NJ, (1989).
5. Mark, W.R., Glanville, R.S., Akeley, K., Kilgard, M.J.: Cg: a system for programming graphics hardware in a C-like language, *ACM Transactions on Graphics.* ACM Press, (2003) 896-907.
6. Moreland, K., Angel, E.: The FFT on a GPU, *Proceedings of the ACM SIGGRAPH Conference on Graphics Hardware*. Eurographics Association, (2003) 112-119.
7. NVIDIA Corporation: GeForce 6800 Product Overview. http://www.nvidia.com (2004).
8. Payne, B.R.: Accelerating Scientific Computation in Bioinformatics by Using Graphics Processing Units as Parallel Vector Processors. (Doctoral dissertation, Georgia State University, 2004). *Dissertation Abstracts International* (UMI. No. pending).

Consistent Spherical Parameterization

Arul Asirvatham[1], Emil Praun[1], and Hugues Hoppe[2]

[1] School of Computing - University of Utah, USA
{arul, emilp}@cs.utah.edu
[2] Microsoft Research, Redmond, USA
http://research.microsoft.com/~hoppe

Abstract. Many applications benefit from surface parameterization, including texture mapping, morphing, remeshing, compression, object recognition, and detail transfer, because processing is easier on the domain than on the original irregular mesh. We present a method for simultaneously parameterizing several genus-0 meshes possibly with boundaries onto a common spherical domain, while ensuring that corresponding user-highlighted features on each of the meshes map to the same domain locations. We obtain visually smooth parameterizations without any cuts, and the constraints enable us to directly associate semantically important features such as animal limbs or facial detail. Our method is robust and works well with either sparse or dense sets of constraints.

1 Introduction

Several applications in computer graphics, such as texture mapping, compression, surface processing, detail synthesis, and object recognition rely on mesh parameterization. Parameterization refers to computing mappings between surfaces in 3D and simpler domains such as planar regions, simplicial domains, or spheres. To be useful for these varied applications, the mappings must satisfy several properties, such as continuity, bijectivity, smoothness, constraint satisfaction, and acceptable distribution of domain area over the surface.

Many parametrization techniques involve partitioning the surface into simpler pieces using cuts. Such cuts are problematic for applications involving a large number of models, since they can lead to fragmentation of the map (i.e. smaller and smaller pieces across which there is a continuous map between all models). Moreover, the resulting map generally lacks smoothness along cuts and at their junctions.

Continuous parameterizations are only possible between topologically equivalent models. For the genus-zero models that we consider, the unit sphere is a natural parameterization domain since it is inherently smooth. Allowing the user to specify constraints is another important requirement in many applications, since it provides a way to incorporate higher-level semantic knowledge about the objects.

We extend the work of Praun and Hoppe [14] on spherical parameterization to allow simultaneous parameterization of multiple objects with point feature constraints while guaranteeing continuity, bijectivity, visual smoothness, and minimizing overall distortion. Using these consistent spherical parameterizations, one can create consistent

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 265 – 272, 2005.
© Springer-Verlag Berlin Heidelberg 2005

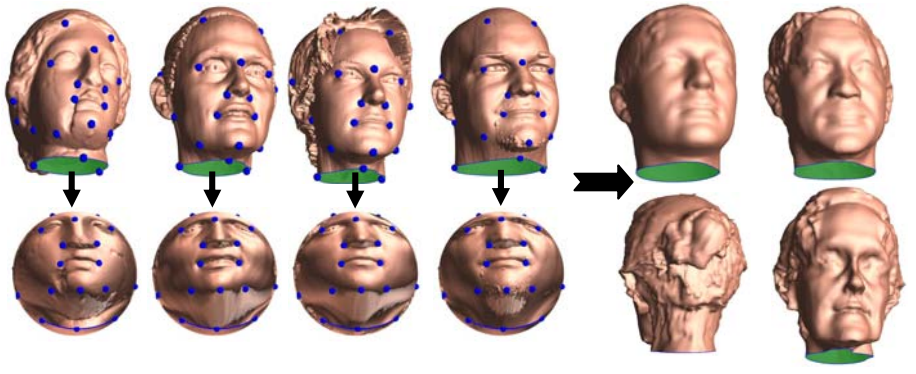


Fig. 1. Consistent Spherical Parameterization of a collection of heads

geometry images [6; 14] representing several objects in a database, opening the door to a large array of applications including compression, conversion to hardware-supported subdivision surfaces with displacement maps, natural LODs, etc.

We present the following contributions:

- Robust construction of consistent spherical parameterizations for several surfaces.
- Constrained spherical parameterization where specified points on a mesh map to given points on the sphere.
- Methods to avoid swirls, and to correct them when they arise.

2 Previous Work

Planar Parameterization: The earliest parameterization methods established mappings to planar domains. There have been many methods developed to date; for a survey we refer the reader to Floater and Hormann [3]. Unless applied to topological discs, these methods must cut the mesh into one or several pieces, introducing discontinuities in the parameterization.

Spherical Parameterization: Discontinuities can be avoided altogether by mapping models to smooth domains of the same topology, such as the unit sphere for genus-zero models. Examples of spherical parameterization methods include [18; 1; 9;7; 5; 14]. We build upon the method of Praun and Hoppe [14].

Incorporating hard constraints into a spherical parameterization scheme proves to be challenging, because it is difficult to guarantee bijectivity of the result. We chose to build upon the approach of Praun and Hoppe [14] because its hierarchical construction approach enables a robust solution.

Parameterization Constraints. Several methods address the problem of parameterization under constraints. Lévy [13] texture maps meshes under soft constraints by introducing additional terms in the quality metric. Eckstein et al. [2] propose a method for satisfying hard constraints for planar parameterization.

Kraevoy et al. [11] start with an unconstrained planar parameterization and then move the constrained vertices to their required positions by matching a triangulation

of these positions to a triangulation of the planar mesh formed by the paths between constrained vertices. Finally, they relax this parameterization while keeping the constraints. They demonstrate results involving a fairly large number of features.

In contrast, our method works well even with few provided feature points. In addition, we use a multiresolution approach to avoid forming an initial parameterization and then optimizing it.

Consistent Parameterization. Praun et al. [15] consistently parameterize a set of genus-0 models onto a user-specified simplicial complex. Unlike their method, we do not impose a fixed consistent connectivity of the base domain, but only satisfy the given point constraints, thereby providing the map more freedom.

Two recent works, by Schreiner et al. [17] and Kraevoy and Sheffer [12], improve upon the technique of Praun et al. by not requiring the simplicial complex to be specified a priori. However, these new techniques do not scale well with regard to the number of models to be consistently parameterized. The technique of Schreiner et al. is limited to dealing with only 2 models, and while Kraevoy and Sheffer demonstrate consistent parameterization among 3 models, their approach is asymmetric and would not scale to a large collection of models.

Whereas in all previous cases the simplicial domain is abstract (with no inherent geometry), our spherical domain geometry is explicit. This implies that paths must be straight arcs on the sphere, rather than arbitrary meandering mesh paths, which makes the problem more difficult. One advantage of the lack of domain connectivity is the opportunity to improve the map by flipping edges, like in Delaunay refinement.

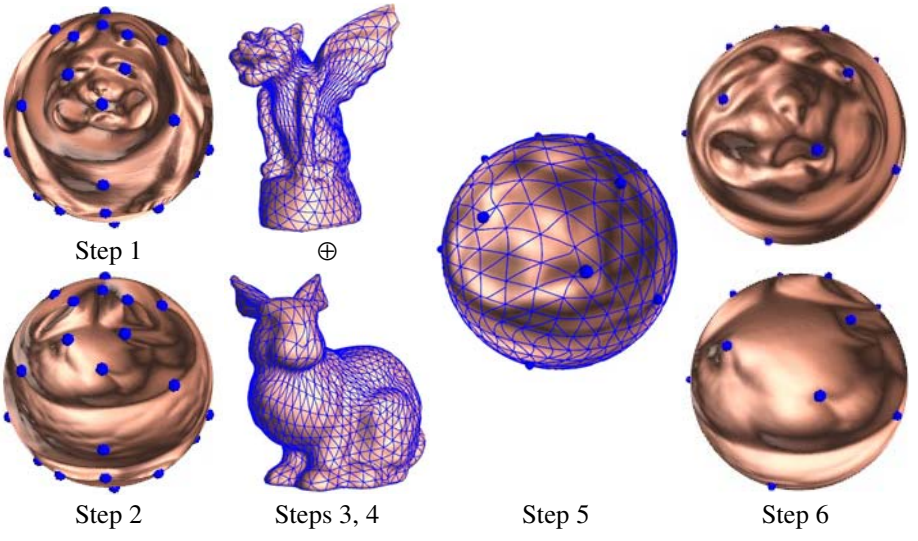


Fig. 2. Steps of the algorithm. For clarity, steps 3-5 show a coarser version of the remesh grid than the one actually used. The features in step 5 are vertices of the finer grid. Stretch efficiencies: gargoyle 0.632, bunny 0.697

3 Approach

Given a set of meshes and corresponding feature points, we form a consistent parameterization of all the meshes. Our approach has two major parts. First, we find good spherical feature locations, such that the final maps have low distortion and distribute the sphere area adequately to the various parts of the input meshes. Second, we create a constrained spherical parameterization for each surface, forcing the feature points to map to the computed locations. Here is the algorithm in more detail (see also Figure 2). Let M_i (i=1..n) be the initial meshes, P be a parameterization, and F a set of spherical feature locations.

1. (P_1', F') := UnconstrainedSphericalParam(M_1)
   //Initial feature locations F' on sphere using one model.
2. For i=2..n, P_i' := ConstrainedSphericalParam(M_i , F')
   //Parameterize all models using those initial locations.
3. For i=1..n, M_i^R := Remesh(M_i , P_i')
   //Remesh to n geometry images with identical connectivities.
4. M_*^R := { M_1^R , M_2^R , … , M_n^R }
   //Concatenate to single mesh with vertex coordinates in R^{3n}.
5. (P^R,F) := UnconstrainedSphericalParam(M_*^R)
   //Find good feature locations considering all models.
6. For i=1..n, P_i := ConstrainedSphericalParam(M_i , F)
   //Compute final parameterizations using these locations.

The main new procedure is the constrained spherical parameterization used in steps 2 and 6. It is described in detail in Section 4. The remaining steps are adapted from earlier work [14]. In step 3 we need to exactly represent the feature points, so we snap the closest grid samples to the spherical locations F'.

Step 5 involves the construction of a progressive mesh for the special mesh M_*^R with geometry in R^{3n}. We modify the quadric error metric [4] to sum each of the n errors in R^3. We also modify the spherical parameterization [14] to sum the n stretch energies from the sphere to the n mesh geometries in R^3.

4 Constrained Spherical Parameterization

We adapt the method of [14] to work with constraints as follows. During coarse-to-fine refinement, we can simply fix the spherical location of feature vertices. The difficulty is to bootstrap the algorithm by creating a valid starting state that satisfies all constraints. Specifically, we must create a progressive mesh representation of the surface where the base domain contains only feature vertices and is triangulated the same way as the spherical features (see Figure 3).

To satisfy this, we need only find a spherical triangulation and a corresponding embedding of its arcs onto the mesh, given by a set of non-intersecting paths. Once we have such a path network, we simplify the mesh to produce a progressive mesh , but we keep the feature vertices, and only allow vertices on the feature paths to be collapsed into other path or feature vertices [16].

Fig. 3. "Triangulations" of features on the sphere (left) and mesh (middle); base mesh after simplification (right)

To produce the path network on the 3D surface, we use a method similar to those of Praun et al. [15] and Kraevoy et al. [11]. We link together pairs of feature points, with great circle arcs on the sphere, and paths on the mesh, until we complete a full "triangulation". The paths (and arcs) cannot intersect each other except at feature vertices (and their spherical locations). The algorithm proceeds in greedy fashion, selecting the best pair from a pool of candidates. To guarantee both termination and topological equivalence of the two "triangulations", we maintain consistent ordering of neighbors around each vertex in the partially completed graph, and we avoid adding any arcs/paths closing cycles before we have linked all the vertices in a spanning tree [15].

The candidate pool is populated initially by shortest paths between all the feature pairs, computed using a Dijkstra search on mesh vertices. When we select the best candidate path we check to see if it intersects any paths already inserted in the network and if it does we re-compute it using a restricted search. These restricted searches can also use edge midpoints in addition to mesh vertices (though with a cost penalty), corresponding to inserting "Steiner" vertices in the original mesh [11].

To improve the geometric quality of the triangulation we employ a set of heuristics to avoid and fix swirls (Section 4.1) and we flip edges by replacing them with the other diagonal of the quad formed by the two adjacent patches. This is done only when the new path is shorter and the new configuration is valid on the sphere.

4.1 Dealing with Swirls

Sometimes the paths appear bad to a human observer because they take unnecessarily long routes around other feature vertices. We call a *swirl* the local configuration of these long paths (Figure 4).

Praun et al. [15] note that swirls cannot be repaired using 1-ring relaxations of patches around a vertex, and propose several heuristics to prevent them. Unfortunately we do not benefit from a user-provided base domain, so some of their heuristics do

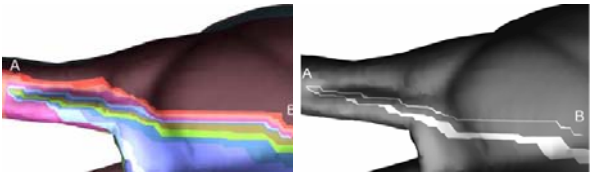


Fig. 4. Swirl on the horse leg: the white patch (right) has to connect to B, but it does so around A. It cannot be straightened sine it would have to move over A and AB

not work in our case. Furthermore, our setting is harder since the spherical locations of the feature points obtained in steps 1 and 5 of the algorithm (Section 3) may be quite different from preferred spherical locations considering the geometry of the current mesh. We develop a more robust set of swirl-avoiding heuristics, as well as a method to remove swirls *after* they have appeared.

Swirl-Avoiding Heuristics. The main tool we use in selecting new paths to insert in the network is their ranking in the priority queue. Paths are ranked according to their surface length (shorter is better), but they are occasionally penalized (placed at the end of the queue) when certain conditions occur. Such conditions include:

- The current path links non-extreme vertices, and there are still some extremities left unconnected. Extremities are features with large average distance to their nearest neighbors (such as legs, arms, etc.). We start the spanning tree construction by linking such features. If left unconnected they might cause swirls since paths linking other vertices go around the base of the extremity, equally likely on the "correct" as on the "wrong" side.
- The spherical image of the path would create spherical triangles with very small angles (<10 degrees in our examples). In these cases the winding order of the 3 feature locations on the sphere is not reliable. Furthermore, skinny triangles make the spherical optimization less robust.
- Failed sidedness tests for neighboring features. We check whether the projection of a feature vertex onto the path is on the same side as the projection of the corresponding feature point onto the arc on the sphere. If some of the vertices are on different sides on the mesh and the sphere, we try to force the path to lie on the correct side of nearby feature vertices. To do this we add temporary constraint paths from the path endpoints to the neighboring feature. We now trace the shortest path on the mesh. The path so traced will be on the correct side of all the features in the connected component of the neighbor. However, this might not always be possible as the addition of the temporary constraint paths might form a cycle enclosing either the source or the destination vertex on different sides on the mesh and the sphere. In such cases, we add the path to the end of the queue.

Unswirl Operator. In addition to using heuristics that avoid swirls (as Praun et al. did [15]), we have also developed a method to identify and remove them in the rare cases that they do occur. Paths between two features incident to many other "long" paths (with high ratios between actual length and geodesic distance between endpoints) are likely to be the center of swirls. To fix them, we remove all paths incident to the two vertices, and then replace them in a new order, introducing first paths that were previously bad.

5 Results and Applications

Figure 1 illustrates the basic approach for applications making use of large model databases. To create the database, a few representative models are selected and consistently parameterized using our algorithm, in order to obtain good spherical locations for the 22 feature points. In our example we used the 5 heads shown out of a set of 8. The remaining models can be subsequently added to the database by running *con-*

strained spherical parameterization. Once the database is created, the consistent parameterization can be used for tasks such as classification and retrieval based on principal component analysis. On the right side of Figure 1 we show the average of our set of heads, and the first three principal components (visualized added to the average head).

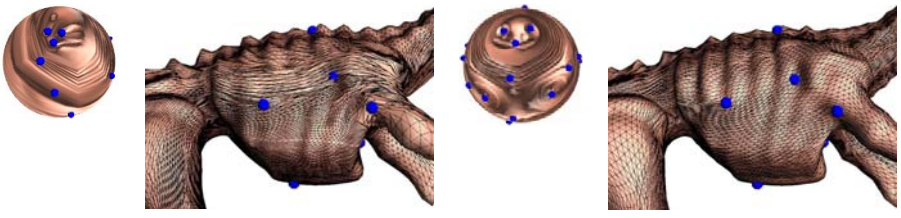


Fig. 5. Parameterization quality improves after optimizing the map taking into account all models

Figure 5 demonstrates the role of steps 3-6 of the algorithm. The two left images show the spherical parameterization and regular remesh using the locations obtained using the cow model alone. The quality improves (right) when the feature locations are computed using all the models.

Consistent parameterizations can also be used to transfer mesh properties, such as geometric detail (the high-frequency components of a multiresolution mesh representation), normals, colors, or texture coordinates. Figure 6 demonstrates examples of color and normals transfer between two heads.

Fig. 6. The texture and normals of the head on the left are combined with the geometry of the head in the middle to produce the one on the right

Table 1. Timing results (in minutes) for the heads example (Figure 1), bunny and gargoyle example (Figure 2). The timings for steps 2 and 6 are cumulative for the different models in the set

	Steps				
Models	1	2	5	6	Total time
Fig. 1	19	81	8	95	203
Fig. 2	10	5	5	17	37

Table 1 shows timing results for our method. For small meshes, step 5 is the most expensive since it involves several geometries simultaneously. However, when the original meshes are dense, step 6 becomes the most expensive one.

6 Summary

We have presented a robust algorithm for parameterizing genus-zero models onto a sphere in the presence of feature constraints. The central part of the algorithm, *constrained* spherical parameterization is guaranteed to produce topologically equivalent spherical triangulations and mesh patch partitions, and avoids awkward *swirl* configu-

rations through a collection of novel heuristics. The regularly sampled consistent geometry images that can be obtained using our parameterizations allow digital geometry processing applicable to many real-world applications.

References

1. Alexa, M. 2000. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1), pp. 26-37.
2. Eckstein, I., Surazhsky, V. and Gotsman, C. 2001. Texture mapping with hard constraints. *Eurographics 2001*, pp. 95-104.
3. Floater, M. and Hormann K. 2004. Recent advances in surface parameterization. *Multiresolution in geometric modeling 2004*.
4. Garland, M. and Heckbert, P. 1997. Surface simplification using quadric error metrics. *ACM SIGGRAPH 97,* pp. 209-216.
5. Gotsman, C., Gu, X. and Sheffer, A. 2003. Fundamentals of spherical parameterization for 3D meshes. *SIGGRAPH 2003,* pp. 358-364.
6. Gu, X., Gortler, S., And Hoppe, H. 2002. Geometry images. *ACM SIGGRAPH 2002,* pp. 356-361.
7. Gu, X., Wang, Y., Chan, T., Thompson, P, and Yau, S.-T. 2003. Genus zero surface conformal mapping and its application to brain surface mapping., *Information Processing Medical Imaging 2003*.
8. Guskov, I., Vidimče, K., Sweldens, W., and Schröder, P. 2000. Normal meshes. *ACM SIGGRAPH 2000,* pp. 95-102.
9. HAKER, S., ANGENENT, S., TANNENBAUM, S., KIKINIS, R., SAPIRO, G., AND HALLE, M. 2000. Conformal surface parameterization for texture mapping. *IEEE TVCG*, 6(2), pp. 181-189.
10. Hoppe, H. 1996. Progressive meshes. *ACM SIGGRAPH 96*, pp. 99-108.
11. Kraevoy, V., Sheffer, A. and Gotsman, C. 2003. Matchmaker: constructing constrained texture maps. *SIGGRAPH 2003,* pp. 326-333.
12. Kraevoy, V., and Sheffer, A. 2004. Cross-parameterization and compatible remeshing of 3D models. *SIGGRAPH 2004,* to appear.
13. Lévy, B. 2001. Constrained texture mapping for polygonal meshes. *ACM SIGGRAPH 2001*, pp. 417-424.
14. Praun, E. and Hoppe, H. 2003. Spherical parameterization and remeshing. *ACM SIGGRAPH 2003,* pp. 340-350.
15. Praun, E., Sweldens, W. and Schröder, P. 2001. Consistent mesh parameterizations. *ACM SIGGRAPH 2001,* pp. 179-184.
16. Sander, P., Snyder, J., Gortler, S., and Hoppe, H. 2001. Texture mapping progressive meshes. *ACM SIGGRAPH 2001*, pp. 409-416.
17. Schreiner, J., Asirvatham, A., Praun, E., and Hoppe, H. 2004. Inter-surface mapping. *ACM SIGGRAPH 2004*.
18. SHAPIRO, A. AND TAL, A. 1998. Polygon realization for shape transformation. *The Visual Computer*, 14 (8-9), pp. 429-444.

Mesh Smoothing via Adaptive Bilateral Filtering

Qibin Hou[1], Li Bai[1], and Yangsheng Wang[2]

[1] School of Computer Science and IT, University of Nottingham,
Nottingham, NG8 1BB, UK
{qbh, bai}@cs.nott.ac.uk,
[2] Institute of Automation, Chinese Academy of Sciences,
Beijing, 100080, China
wys@nlpr.ia.ac.cn

Abstract. In this paper, we present an adaptive bilateral filtering algorithm that can be used to remove unavoidable noise from 3D mesh data generated by initial stages. Selecting the parameters for bilateral filters automatically, this algorithm smoothes meshes in the normal field using anisotropic character of local neighborhood triangles. Experimental results demonstrate that the proposed method remove light noise from meshes and reserve fine features of meshes as good as best results of other methods, with the advantage of none user-assisted parameters setting. Visual comparisons display that the method proposed in this paper performs better than other smoothing method for heavy noisy mesh.

1 Introduce

Over recent years, 3D computer graphics research results have been applied in interactive entertainment industry to meet the public's increasing thirst for fun at an enormous rate. In many applications, 3D mesh processing act an important role for visual representation of 3D content as realistic as possible. This depends on the quality of 3D data which is created by scanning or modelling process. However, the raw geometry data acquired from the real-world object usually contains undesirable noise from various sources, even with high fidelity scanners [1]. Therefore, smoothing is necessary before further stages, with the purpose to remove noise while preserving the crucial shape features.

Amount of mesh smoothing or mesh denoising algorithm have been published so far, for example [2], [3], [4], [5], [6], [7], [8], [9], [10]. Taubin [2] put forward a simple, linear and isotropic method to enhance the smoothness of triangulated surfaces at first. After that, Desbrun et al [3] made an improvement by using a geometric flow analogy for irregular meshes. Inspired by image processing work on scale space and anisotropic diffusion, Zhang [7] and Bajaj [10] proposed different feature-preserving surface fairing method. Ohtake [4], Taubin [5], Tasdizen [6] offered diffusion-type smoothing on the normal field itself. Jones [9] analyzed the advantages and disadvantages of all these methods, and achieved impressive results based on robust statistics and local first-order predictors of the surface.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 273–280, 2005.
© Springer-Verlag Berlin Heidelberg 2005

He utilized a method analogous to bilateral filtering for images. Contemporaneously, Fleishman [8] presented a similar method with Jones that uses bilateral filtering for smoothing surfaces, but exploited a different distance way. However, both the methods exploited by Jones [9] and Fleishman [8] are needed to choose parameters assisted by user.

By integrating Ohtake [4] with Fleishman [8] together, this paper attempts to present an idea of developing self-adaptive parameters setting bilateral filter for noisy triangle meshes. The aim is to automatically smooth a noisy mesh and preserve sharp features without help from user. The rest of this paper is organized as follows: Section 2 describes background about bilateral filtering for images, and adaptive Gaussian filter, and how to adapt and extend them for meshes. Section 3 describes our adaptive bilateral filter smoothing method. Experiment results are shown in Section 4. Section 5 has conclusion and proposal for future work.

2 Background About Bilateral Filter and Adaptive Gaussian Filter

2.1 Bilateral Filter for Images and Its Extension to Meshes

Bilateral filter is proposed by Tomasi et al. [11] for edge-preserving image smoothing. For image I, the formulation to calculate the output value at pixel $u(x, y)$ is [11]:

$$E(u) = \frac{\sum_{p \in N(u)} G(\|p-u\|) S(|I(p)-I(u)|) I(p)}{\sum_{p \in N(u)} G(\|p-u\|) S(|I(p)-I(u)|)} \tag{1}$$

where $N(u)$ is the neighbourhood of u. $G(\|p-u\|)$ measures the geometric closeness, $S(|I(p)-I(u)|)$ measures the photometric similarity. It can be seen that $E(u)$ depends not only on the spatial distance $\|p-u\|$, but also on the signal difference $|I(p)-I(u)|$.

In order to filter a mesh triangle T using local neighboring triangles T_j, we try to extent Formula (1) for processing mesh normals. The triangle normals can be seemed as vector-valued image defined over the meshes. Let $n(T)$ is the normal value at triangle T, C is the centroid of T, $n(T_j)$ is the normal value at triangles T_j, and C_j is the centroid of T_j. Spatial weight G depends on the geodesic distance $d(C, C_j)$. $d(C, C_j)$ is found via Dijkstra's algorithm on the dual mesh which is formed by the centroids [4]. Influence weight S depends on the difference between $n(T)$ and $n(T_j)$, which is denoted by $h(T, T_j) = \|n(T) - n(T_j)\|$.

The smoothed normal $m(T)$ is computed by

$$m(T) = \frac{\sum A(T_j) G(d) S(h) n(T_j)}{|\sum A(T_j) G(d) S(h) n(T_j)|} \tag{2}$$

where $A(T_j)$ is the area of T_j.

According to Jones [9], the spatial weight and the influence weight are able to use Gaussian filter. By selecting width of σ_g of the spatial and σ_s of the influence weight Gaussians, the amount of smoothing is achieved.

Mesh normal of triangle T due a Bilateral filtering can also be described as:

$$n(T) = m_{\sigma_g,\sigma_s}(T) + \varepsilon_{\sigma_g,\sigma_s}(T) \tag{3}$$

where $\varepsilon_{\sigma_g,\sigma_s}$ is the difference between original normal and the smoothed normal.

2.2 Adaptive Gaussian Filter for Local Smoothness

Gomez [12] gave an adaptive Gaussian filter in which local variance σ for each kernel is selected automatically by working with a scale-space framework and minimal description length criterion (MDL).

The filtering procedure can be understood as [12]:

$$I_0(x,y) = I_\sigma(x,y) + \varepsilon_\sigma(x,y) \tag{4}$$

where $I_\sigma(x,y)$ is the Low-Pass part, and $\varepsilon_\sigma(x,y)$ is the residual part. Rewrite Equation (4) as description length (dl), [12] has following equation:

$$dl_{I_0} = (\frac{\gamma}{\sigma_x^2}) + \varepsilon^2, \gamma = \frac{2n\sigma_\varepsilon^2\alpha\beta}{k} \tag{5}$$

where $n \geq 2, \alpha, \beta$ are given constants.

An image smoothed by adaptive Gaussian filter is [12]:

$$I_\sigma(x,y) = I_0(x,y)exp(-\frac{x^2+y^2}{2\sigma^*(x,y)^2}) \tag{6}$$

where $\sigma^*(x,y)$ is local variance at selected scale from $\sigma_1, \sigma_2, \ldots, \sigma_n$. At that scale, description length of I_σ has minimal value at location (x,y).

3 Adaptive Bilateral Mesh Smoothing

In our proposed method, bilateral filter and adaptive parameter selection are combining together for smoothing the normal fields. And then, fitting meshes to modified normals to achieve smoothed results.

3.1 Normal Smoothing Algorithm Description

Define Ω is composed by the σ_{min}, σ_{max}, and step length λ, these three variances are decided by average edge length of all the meshes. Please refer to Equation (2) for means of other variances, for example, n is the normal of T, C is the centroid of T, the neighborhoods of T denoted by $\{T_j\}$, etc. The pseudo-code for applying Bilateral filter to triangle normal with parameter automatic setting is as follows:

Name: Adaptive Bilateral Normal Smoothing

Input: $\{T, n, \Omega | T = (v_1, v_2, v_3); n(T) = (x, y, z); \Omega = (\sigma_{min}, \sigma_{max}, \lambda)\}$

Output: $m(T) = (x', y', z')$

Step 1: Search for neighborhoods $\{T_j\}$, which meet the requirement of $D(C_j, C) \leq 2\sigma_{max}, j = 1, \ldots, k$

Step 2: Calculate influence weight parameter and Gaussian filtering
$S(h) = exp(-\frac{h^2}{2\sigma_s^2})$, where $\sigma_s = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(h_j - \hbar)}$,
$\hbar = \frac{1}{k}\sum_{j=1}^{k} h_j, h_j = \|n(T) - n(T_j)\|$

Step 3: Estimate spatial weight parameter with MDL at scale space

FOR $j = 1 : 1 : k$

FOR $\sigma_i = \sigma_{min} : \lambda : \sigma_{max}$
$G(d) = exp(-\frac{d^2}{(2\sigma_i^2)})$
$m_{\sigma_i,\sigma_s}(T) = \frac{\Sigma A(T_j)G(d)S(h)n(T_j)}{|\Sigma A(T_j)G(d)S(h)n(T_j)|}$
$\varepsilon^2 = 2(1 - |\frac{\sum A(T_j)G(d)S(h)n(T_j)}{\sum A(T_j)G(d)S(h)}|)$

END

END

THEN $m_{\sigma_g,\sigma_s}(T) = \{m_{\sigma_i,\sigma_s}(T) | \sigma_g = [argmin(\frac{\mu}{\sigma_i^2}) + \varepsilon^2]\}$

Step 4: Get smoothing result, normalize the average normal $m(T) = m(T)/\|m(T)\|$

3.2 Mesh Fitting Based on Modified Normals

From Sec. 3.1, the smoothed normal for each triangle has been obtained. The further need to do is to rebuild mesh from the smoothed normals. Ohtake [4] gave an efficient method to modify the mesh vertex positions for fitting mesh.

For each mesh vertex V, [4] define

$$E_{fit}(V) = \sum_{j=1}^{n} A(T_j)|n(T_j) - m(T_j)|^2 \tag{7}$$

where n is the amount of all triangles that are adjacent to V.

Then new position for V is

$$V_{new} = argmin(E_{fit}(V)) \tag{8}$$

A simple conjugate gradient method [13] is used to explore the minimum of $E_{fit}(V)$. Please refer to [4] for detail implementation.

4 Results and Comparison

We realized the adaptive bilateral mesh smoothing methods described in last section based on the platform provided by Ohtake. Fig. 1(c) demonstrates our result, comparing with the result in Fig. 1(b) from Ohtake's adaptive Gaussian and anisotropic neighborhood [4].

In all cases of this faction, Gaussians are used for spatial (G) and influence (S) weight with standard deviation of σ_g, σ_s individually. Other robust weight functions can be applied as well. Jones [9] found Gaussian has good performance in those two weights. In Fig. 1, the choice for σ_s depends on Gaussian error normal of neighborhoods, which were selected with the help of σ_{max}, the maximum value of σ_g. To find σ_g, we use ten scale-spaces with σ from $\sigma_{min} = 0.4l$ to $\sigma_{max} = 4.0l$ with step size $\lambda = 0.4l$, something like what Ohtake have done in [4]. l is the arithmetic mean of the edge lengths of the mesh. For each mesh triangle, local values of σ_g, σ_s are set by algorithm adaptively, not by user. From Fig. 1, we can see that our method makes same good smoothing as that best result produced by Ohtake.

It is known that isotropic methods indiscriminately smooth noise and outstanding features at same time [9]. Many researchers have proposed various anisotropic method to overcome this limitation of isotropic smooth technique. Durand and Dorsey [14] show that bilateral filtering is alike with anisotropic diffusion. This important character of bilateral filtering helps it to remove noise while preserving the shape. A comparison of Taubin's $\lambda|\mu$ smoothing method [5] with our adaptive bilateral filtering scheme, is presented by Fig. 2.

From Fig. 2, the vision result of Fig. 2(c) seems smoother at general surface of face than Fig. 2(b), while Fig. 2(g) keeps the eyes of model more eminent than Fig. 2(e). It shows that outstanding features such as eyes were kept, while noises were removed efficiently. This demonstrates the effectiveness of our method at feature preservation.

Fig. 1. Results of smoothing Stanford Bunny with word 'iH' embossed. (a) Original mesh. (b) Best visual result obtained with Ohtake's adaptive Gaussian + anisotropic neighborhood. (c) Adaptive bilateral smoothing proposed in this paper. The model is courtesy of A. Belyaev

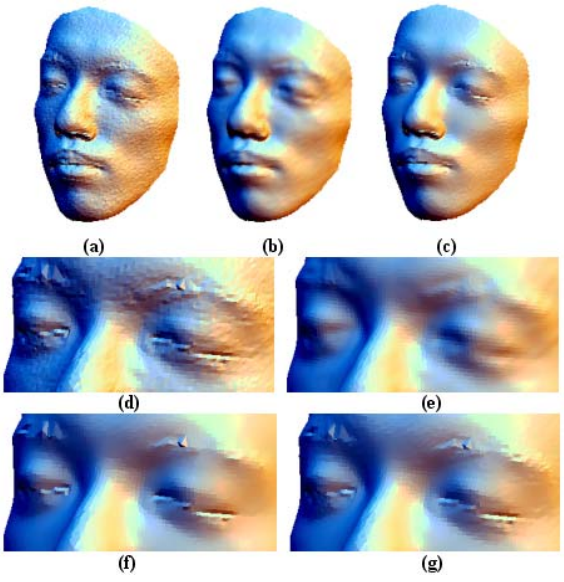


Fig. 2. Results of smoothing 3D face model. (a) Original face mesh. (b) Result of Taubin's $\lambda|\mu$ smoothing method. (c) Result of our method. (d) Zoom the eyes of original model. (e) Zoom the eyes of result from method of Taubin. (f) Zoom the eyes of result from method of Ohtake. (g) Zoom the eyes of result from our method. The model is courtesy of A. Belyaev

Ohtake [4] said that one of drawbacks of his method is not able to remove large noise. We test the capability of our method by smoothing mesh at different noise level in Fig. 3. Fig. 3(a) is a monk mesh obtained using laser scanner. Fig. 3(a) is smoothed by adaptive Gaussian and anisotropic neighborhoods, which produces Fig. 3(b). Fig. 3(c) is our own result. Fig. 3(c) and Fig. 3(b) have similar visual quality. This means that both our method and Ohtake's method are valuable for light noisy mesh. However, when they cope with heavy noisy mesh, there will be a little different effect, as shown in Fig. 3(e) and Fig. 3(f) individually. Fig. 3(d) is obtained by adding heavy noise to Fig. 3(a). Fig. 3(e) is the result of Ohtake's method used to smooth Fig. 3(d). Fig. 3(f) is got by adaptive bilateral filter. A slight improvement of visual quality is achieved in Fig. 3(f) comparing with Fig. 3(e). This result shows the capability of our method to smooth mesh in the presence of little heavy noise.

5 Conclusion

In this paper, we propose adaptive bilateral mesh filtering method in which two parameters are set by fully automatic solution according to the shape of local triangle neighborhoods. This improved method adaptively smoothes light noisy mesh and keep the sharp features as well, with ideal human visual effect. In

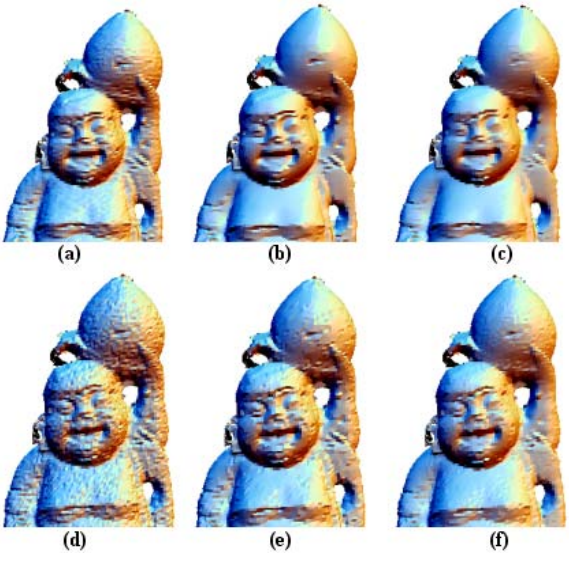


Fig. 3. Results of smoothing at various noise level. (a) Original monk mesh. (b) Smoothed by Ohtake's Gaussian + anisotropic method. (c) Adaptive bilateral filter was applied. (d) Noisy monk mesh. (e) Ohtake's method for noisy mesh. (f) Our method for noisy mesh

addition, it can also work for large noisy mesh with a little better robustness, and outperforms other methods in a way.

This adaptive bilateral mesh smoothing method is based on iteration. So the time-consuming is the drawback of the developed method. To find avenue for fixing this weakness will be a task for future work. Another work need to do is to establish a general quality evaluation rule or metrics for 3D mesh smoothing. That metrics should be consistent with human perception of 3D shape. This is another important theme for future work, as Belyaev has said [15].

Acknowledge. Special thanks to Yutaka Ohtake for providing basic platform on which we can implement our algorithm and comparison efficiently. Original meshes in this paper courtesy of A. Belyaev from his homepage. This work was funded by University of Nottingham (Project Code A2RH04). We would like to thank the anonymous reviewers of this paper for their valuable comments.

References

1. Rusinkiewicz, S., Hall-Holt, O., and Levoy, M.: Real Time 3D Model Acquisition. ACM Trans. Graphics 21, 3, 438-446
2. Taubin, G.: A Signal Processing Approach to Fair Surface Design. In Processings of SIGGRAPH 1995, 351-358
3. Desbrun, M., Meyer, M., Schroder, P., and Barr, A. H.: Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. Im Proceedings of SIGGRAPH 1999, 317-324

4. Ohtake, Y., Belyaev, A. G., and Seidel, H. P.: Mesh Smoothing by Adaptive and Anisotropic Gaussian Filter. In Vision, Modeling, and Visualization 2002, 203-210, Erlangen, Germany, November 2002.
5. Taubin, G.: Linear Anisotropic Mesh Filtering. IBM Research Report RC22213 (W0110-051), IBM, October 2001
6. Tasdizen, T., Whitaker, R., Burchard, P., and Osher, S.: Geometric Surface Smoothing via Anisotropic Diffusioin of Normals. In Proceedings, IEEE Visualization 2002, 125-132
7. Zhang, H., and Fiume, E. L.: Mesh Smoothing with Shape or Feature Preservation. In Advances in Modeling, Animatioin, and Rendering, J. Vince and R. Earnshaw, editors, 167-182, 2002
8. Fleishman, S., Drori, I., and Cohen-Or, D.: Bilateral Mesh Denoising. Proceedings of ACM SIGGRAPH 2003
9. Jones, T., Durand, F., and Desbrun, M.: Non-iterative, Feature-preserving Mesh Smoothing. ACM Trans. Graphics 2003, 22, 3, 943-949
10. Bajaj, C., and Xu, G.: Anisotropic Diffusion on Surfaces and Functions on Surfaces. ACM Trans. Graphics 2003, 22, 1, 4-32.
11. Tomasi, C., and Manduchi, R.: Bilateral Filtering for Gray and Color Images. In Proc. IEEE Int. Conf. On Computer Vision, 1998, 836-846
12. Gomez, G.: Local Smoothness in Terms of Variance: the Adaptive Gaussian Filter. In Proc. Of the BMVC 2000, vol.2, 815-824
13. Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P.: Numerical Recipies in C: The Art of Scientific Computing. Cambridge University Press, 1993.
14. Durand, F., and Norsey, J.: Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. ACM Trans. Graphics 2002, 21, 3, 257-266
15. Belyaev, A., and Ohtake, Y.: A Comparison of Mesh Smoothing Methods. Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics Tel-Aviv, Feb. 2003, 83-87

Towards a Bayesian Approach to Robust Finding Correspondences in Multiple View Geometry Environments

Cristian Canton-Ferrer, Josep R. Casas, and Montse Pardàs *

Image Processing Group, Technical University of Catalonia,
Barcelona, Spain
{ccanton, josep, montse}@gps.tsc.upc.es

Abstract. This paper presents a new Bayesian approach to the problem of finding correspondences of moving objects in a multiple calibrated camera environment. Moving objects are detected and segmented in multiple cameras using a background learning technique. A Point Based Feature (PBF) of each foreground region is extracted, in our case, the top. This features will be the support to establish the correspondences. A reliable, efficient and fast computable distance, the *symmetric epipolar distance*, is proposed to measure the closeness of sets of points belonging to different views. Finally, matching the features from different cameras originating from the same object is achieved by selecting the most likely PBF in each view under a Bayesian framework. Results are provided showing the effectiveness of the proposed algorithm even in case of severe occlusions or with incorrectly segmented foreground regions.

1 Introduction

Multi camera systems are being widely used for image and video analysis tasks in SmartRooms, surveillance, body analysis or computer graphics. Multiple view geometry has been addressed in [4, 9] from a mathematical viewpoint, but there is still work to be done for the efficient fussion of redundant camera views and its combination with image analysis techniques for object detection and tracking. In this framework, the current paper addresses the problem of finding meaningful correspondences between regions in different views.

In multiple camera environments, given an image sequence, the problem is to find the correspondences among feature points across the images that represent the projection of the same object in the real world in different camera views. Once these correspondences among all feature points are available, they can be used for object tracking and identification, motion analysis and scene analysis.

* This material is based upon work partially supported by the IST programme of the EU through the NoE IST-2000-32795 SCHEMA and IP IST-2004-506909 CHIL and by TEC2004-01914 project of the Spanish Government.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 281–289, 2005.
© Springer-Verlag Berlin Heidelberg 2005

(a) (b)

Fig. 1. In (a) there is an example frame from a multiview camera environment: SmartRoom at UPC. In (b) the result of a foreground segmentation and extraction of labeled blobs

The output of a multiview correspondence algorithm is a set of links, ideally representing how a unique point or an object in the real world corresponds to a specific position in each projected image. For applications like particle tracking or 3D tracking of a number of objects in a cluttered scene with strong occlusions, the redundancy in the different projections should help to overcome occlusion problems.

For a given frame in the video sequence, a set of N images are obtained from the N cameras. Each camera is modeled using a pinhole camera model based on perspective projection. Accurate camera calibration information is available. We use a segmentation algorithm by [10] based on Stauffer-Grimson's background learning and substraction technique [12]. In spite of the good performance of this technique, objects in the scene are oversegmented due to ilumination changes, reflexions and other distorting elements typical in non-controlled environments. Nevertheless, our algorithm proposes solutions to overcome oversegmentation problems and still finds accurate correspondencies.

Once foreground regions are extracted, a labeling process is performed based on connectivity rules, thus obtaining a set of enumerated disjoint foreground regions. Let us denote $\mathcal{B}_k^i(x, y)$, the set of pixels belonging to the k-th labeled foreground region at the i-th camera view and denote this set as blob. The work presented in this paper addresses the problem of establishing the correspondences among the set of blobs representing the projection of the same object in the different cameras. Once these correspondences are decided, this information can be used by further video processing modules such as the initialization of a multiocular tracker [6, 11], body gesture analysis [8] or 3D reconstruction [3].

There are two contributions in this paper. First, the application of a fast and simple cross-view point-to-point distance based on epipolar geometry, the *symmetric epipolar distance*, for guessing possible correspondences among views.

Then, a probabilistic approach to select the right geometric correspondences among blobs present in different views. This method takes into account occlusions and solves them by exploiting redundant information in multiple views.

2 Multiple Camera Object Correspondence

In order to compute candidate 3D locations of objects, an algorithm that guesses which foreground regions from different cameras belong to the same object is required. This grouping of foreground regions is depicted as a correspondence guess. In order to tell valid correspondence guesses from invalid guesses, a Point Based Feature (PBF) from each foreground region is extracted. A cross-view point-to-point distance is introduced to evaluate the validity of each correspondence guess. Correspondence hypotesis are generated by a sequential minimization procedure and, finally, a Bayesian decision algorithm evaluates the correspondence candidates in order to decide whether they are valid or not.

Existing correspondence algorithms [6, 11] rely on nonlinear least-square solutions or overdetermined linear systems. However, by increasing the views (cameras) or the number of detected objects, these systems turn out to be increasingly computationally expensive and, therefore, not appropriate for real-time systems. Other methods [1, 9] rely on homographies between images but, for wide angle lense cameras (such as the SmartRoom's case) homographies present significant distortion.

2.1 Symmetric Epipolar Distance

In order to find correspondences among blobs in different views, a distance that measures how close are the points of a set of points $\mathcal{D}(p) = \{\mathbf{x}^0, \mathbf{x}^1, ..., \mathbf{x}^{p-1}\}$ in p different views must be defined. In this case, closeness is understood as how well these points fit as projections of the same 3D point. This distance should be accurate, reliable and fast to compute. Under these conditions, *symmetric epipolar distance*, a ℓ_2-distance between points in different views based on epipolar geometry is employed. This distance was used in [14] to compare fundamental matrix estimations and has proven to be effective for our purposes. This distance is presented for the case of two points in two different views and afterwards extended to the general case.

Let $l\left(\mathbf{x}^i, j\right)$ be the epipolar line generated by the point $\mathbf{x}^i$ onto view j. We define the symmetric epipolar distance between two points $d_{\mathcal{SE}}(\mathbf{x}^i, \mathbf{x}^j)$, in the two views i,j, as:

$$d_{\mathcal{SE}}(\mathbf{x}^i, \mathbf{x}^j) \triangleq \sqrt{d^2(l(\mathbf{x}^i, j), \mathbf{x}^j) + d^2(l(\mathbf{x}^j, i), \mathbf{x}^i)}, \tag{1}$$

where $d(l(\mathbf{x}^i, j), \mathbf{x}^j)$ is defined as the Euclidean distance between the epipolar line $l\left(\mathbf{x}^i, j\right)$ and the point $\mathbf{x}^j$ as depicted in Fig.2. It can be shown that the extension of the symmetric epipolar distance for $p \geq 2$ points (in p different views) $d_{\mathcal{SE}}(\mathbf{x}^0, \mathbf{x}^1, ..., \mathbf{x}^{p-1})$ can be written in terms of the distance defined in Eq.1 as:

$$d_{\mathcal{SE}}(\mathbf{x}^0, \mathbf{x}^1, ..., \mathbf{x}^{p-1}) \equiv d_{\mathcal{SE}}(\mathcal{D}(p)) = \sum_{i=0}^{p-2} \sum_{j=i+1}^{p-1} d_{\mathcal{SE}}(\mathbf{x}^i, \mathbf{x}^j). \tag{2}$$

Hence, if the set $\mathcal{D}(p)$ is formed by projections of the same location in the 3D world, the distance between them should be very small (ideally zero).

The main advantage of this distance is its simplicity in contraposition with other distances where inverse matrix computations or homography estimation are required [9, 11]. However, despite this distance does not give a meaningful measure of a physical distance, it turns out to deal successfully with multiple view ($p \geq 3$) correspondence problems as explained in Sect. 2.3. When $p < 3$, two 3D points lying in the same epipolar plane would result in a small distance, hence the necessity of $p \geq 3$ in order to resolve the ambiguity.

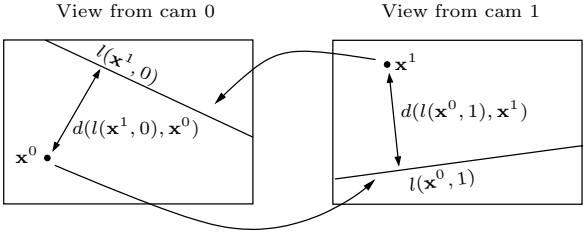


Fig. 2. Symmetric epipolar distance between two points $d(\mathbf{x}^0, \mathbf{x}^1)$

2.2 Point Based Features (PBF)

Point based features are those that only take into account a 3D point and the projection of this point in the existing views. To establish correspondences among blobs in different views, the centroids of blobs have been extensively used as a feature [6, 7, 11]. In the case of SmartRooms and cluttered scenes in general, blobs are affected by severe oclusions (due to furniture or other moving objects), hence their centroids are biased and not suitable as a correspondence feature. The tops of blobs in each view defined as

$$\mathbf{t}_k^i = \frac{1}{2}\left(\min_x \min_y \mathcal{B}_k^i(x, y) + \max_x \min_y \mathcal{B}_k^i(x, y)\right), \tag{3}$$

with i and k being the blob and camera index respectively, have been used as our PBF. It should be noted that the number of PBFs detected in one view may be different from another view.

In contraposition with other features, blob tops represent a meaningful point of the object and once correct correspondences are found, it can be used for forecoming analysis modules of the system (for example, estimating the height of a person). However, it must be said that for cameras positioned in a very low position, tops can be slightly biased due to the lack of visibility of the real top.

2.3 Finding Correspondence Candidates

Once a distance is defined, a method for finding and evaluating candidate correspondences between PBFs in different views is straightforward. Furthermore, if correct point correspondences can be found, blob correspondences across views are derived as well.

Let us suppose that a 3D point $\mathbf{X}$, has projections $\mathbf{x}^i$ on all the N cameras of our system. Despite this assumption is quite strong, let us define a method to find the correspondence guess and, afterwards improve it to deal with less restrictive assumptions. Then, we must choose the set $\mathcal{D}^\star(N)$ that accomplishes

$$\mathcal{D}^\star(N) = \min_{\mathcal{D}(N)} d_{\mathcal{SE}}(\{\mathcal{D}(N)\}), \tag{4}$$

where $\{\mathcal{D}(N)\}$ is the set of all possible correspondence candidates, that is all PBF combinations across all views.This process can be acchieved by either browsing all the possible combinations or reducing the search space with a priori geometric or probabilistic constraints [13].

In the case where multiple targets are present, M correspondence guesses should be determined. In this case, a sequential minimization procedure has been proposed according to [2]. This process is done by selecting the set $\mathcal{D}^\star(N)$ that minimizes the distance at a given iteration and then discarding the N points chosen for the next step, thus decreasing the search space dimension. This process is executed until M sets $\mathcal{D}_k^\star(N)$, $1 \leq k \leq M$, are selected. Results for this technique applied to find the correspondences between $N = 4$ views with $M = 4$ moving objects in a SmartRoom are shown in Fig.3.

In contraposition with the former method, a global minimization method to find the set $\mathcal{D}_k^\star(N)$ fulfilling the criteria

$$\mathcal{D}_k^\star(N) = \min_{\mathcal{D}_k^\star(N)} \sum_{k=0}^{M-1} d_{\mathcal{SE}}(\mathcal{D}_k^\star(N)), \tag{5}$$

has also been studied. That is to find the M sets $\mathcal{D}_k^\star(N)$ that minimize the total distance. Unfortunately, this problem turns out to be analogous to the overcomplete representation problem [5] and therefore an intractable NP-hard problem.

In the case when the 3D feature $\mathbf{X}$ does not have a projection in one or some of the views, our algorithm can still produce correspondence guesses if there are at list 3 projections of $\mathbf{X}$ available. We can find sets $\mathcal{D}^\star(p)$, $3 \leq p \leq N$ by allowing our minimization algorithm to build up the correspondence candidates $\mathcal{D}(p)$ including a null-projection in one or some views. Once a correspondence $\mathcal{D}^\star(p)$ is found, an estimation $\tilde{\mathbf{X}}$ of the 3D position of feature $\mathbf{X}$ can be estimated with a joint estimation back-projected ray with outlier rejection method [4]. By projecting back $\tilde{\mathbf{X}}$ onto all the camera planes we can obtain the set $\bar{\mathcal{D}}^\star(N)$ that completes the missing projections of $\mathcal{D}^\star(p)$ in the case where $p < N$. Moreover, since the information contributed by all the views is taken into account in the estimation $\tilde{\mathbf{X}}$, this new set $\bar{\mathcal{D}}^\star(N)$ refines the original one diminishing local errors.

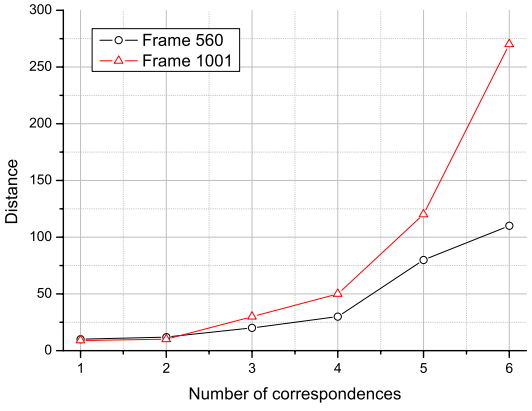


Fig. 3. Distance vs. Number of points selected with $N = 4$ and $M = 4$. The first 4 correspondences present a small and similar distance but, since the fifth corresponcence and following do not belong to any 3D point, their distances increase noticeably

2.4 Bayesian Correspondence Algorithm

The method proposed in the former subsection is able to produce M correspondence guesses for a given number of cameras p. However, since these parameters have to be selected manually, an adaptive algorithm that guesses the number of existing valid correspondences M and selects the optimum number of views p for each correspondence is required.

As a first step of this algorithm, it must be determined whether a correspondence $\mathcal{D}^\star(p)$ is formed by projections of the same 3D feature, denoting this property as belonging to the set $\mathcal{V}$, namely $\mathcal{D}^\star(p) \in \mathcal{V}$. For this task a Bayesian approach is employed. The posteriori probability of $\mathcal{D}^\star(p) \in \mathcal{V}$ given its symmetric epipolar distance $d_{\mathcal{SE}}(\mathcal{D}^\star(p))$ is formally:

$$P(\mathcal{D}^\star(p) \in \mathcal{V} | d_{\mathcal{SE}}(\mathcal{D}^\star(p)) = \frac{P(d_{\mathcal{SE}}(\mathcal{D}^\star(p)) | \mathcal{D}^\star(p) \in \mathcal{V}) P(\mathcal{D}^\star(p) \in \mathcal{V})}{P(d_{\mathcal{SE}}(\mathcal{D}^\star(p))}. \quad (6)$$

Since the priors $P(\mathcal{D}^\star(p) \in \mathcal{V})$ and $P(d_{\mathcal{SE}}(\mathcal{D}^\star(p))$ are wide and uninformative, Eq.6 can be rewritten as:

$$P(\mathcal{D}^\star(p) \in \mathcal{V} | d_{\mathcal{SE}}(\mathcal{D}^\star(p)) \propto P(d_{\mathcal{SE}}(\mathcal{D}^\star(p)) | \mathcal{D}^\star(p) \in \mathcal{V}), \quad (7)$$

where the probability $P(d_{\mathcal{SE}}(\mathcal{D}^\star(p)) | \mathcal{D}^\star(p) \in \mathcal{V})$ is modeled as a Gaussian distribution $\mathcal{N}(m_p, \sigma_p)$. Parameters m_p and σ_p are tunable depending on the accuracy of the segmentation algorithms and the geometry of the environment. This assumption is indeed valid after our observation of the average distribution of the distance $d_{\mathcal{SE}}(\mathcal{D}^\star(p))$. For each value of p, this distribution will be different, obbeying the rule $m_p \geq m_{p-1}$ since the less views are taken into account for the correspondence of the projection of $\mathbf{X}$, the less the overall distance. Conversely,

the less the views employed, the less accurate the estimation $\tilde{\mathbf{X}}$ results. Hence, correspondences with the most views will be preferred. As a direct aplication of this theory, the implementation of our algorithm is based on the sequential selection of the sets $\mathcal{D}^{\star}(p)$ with larger p and higher probability in decreasing order. The algorithm stops when the probability is under a certain threshold α (for our experiments $\alpha > 0.7$).

3 Results

In order to evaluate the performance of the proposed algorithm, we used it to set correspondences in a SmartRoom equiped with 4 fully calibrated cameras where foreground blobs represented people. A sequence of 2000 frames was recorded simulating a presentation meeting with 4 atendees.

The images were segmented but, due to oversegmentation, we obtained more blobs per image than the number of persons. Consequently, the possible combinations grew up to an average of 20000 combinations per set of images. Even in this harsh conditions, the algorithm was able to perform effectively. For comparison purposes, Table 1 shows the performance of our algorithm for two types of PBFs, tops and centroids, where tops outperformed the existing algorithms based on centroids. Fig. 4 illustrates successful correspondences established among blobs representing people in a SmartRoom.

Table 1. Results of correct match and confidence ($\bar{P}$) of the correspondence estimation obtained by applying our correspondence algorithm on 200 frames chosen at random over a sequence of 2000

	Correct Match (%) PBF top	$\bar{P}$ (%) PBF top	Correct Match(%) PBF centroid	$\bar{P}$ (%) PBF centroid
Person 1	99.8	99.2	11.76	21.3
Person 2	99.77	98.62	41.18	43.27
Person 3	96.04	91.3	50.78	37.5
Person 4	95.8	85.6	44.31	29.0

4 Conclusions and Future Work

We have presented an algorithm for finding correspondences between regions in a multiple camera environment. The algorithm employs a probabilistic criterion for matching PBFs in different cameras, in our case the top of the blobs, and has proven very reliable, outperforming the existing approaches to this problem. The results of the algorithm are interesting for the initialization a multiocular tracking of multiple people system, multiview face or body detection, analysis and recognition.

Experimental results verify that we can obtain good quantitative 3D parameters from 2D image observations of people in the scene. We have demonstrated

(a) (b)

Fig. 4. Two examples of the application of our correspondence algorithm. Figure (a) shows its utility for multi-view face detection where the faces of the four people are being localized in all views providing reliable data for higher level analysis. In (b) correspondences are being correctly established in the framework of body analysis

that correspondences between blobs belonging to the same physical object can be found even if the object does not have a projection in some of the views. Future research perspectives involve the development of robust 2D/3D tracking of this PBF correspondences.

References

1. Black, J., Ellis, T.: Multi Camera Image Tracking. In *Proc. IEEE Work. on Performance Evaluation of Tracking and Surveillance*, 2001.
2. Boyd, S., Vandenberghe, L.: Convex Optimization. 1st edn. Cambridge University Press, 2004.
3. Eisert, P., Steinbach, E., Girod, B.: Multi-hypothesis, volumetric reconstruction of 3D objects from multiple calibrated camera views. In *Proc. IEEE Int. Conf. on Acoustics Speech and Signal Processing*, pp. 3509–3512, 1999.
4. Faugeras, O., Luong, Q.T.: The geometry of multiple views. 1st edn. MIT Press, 2001.
5. Figueras, R.M.: Image Coding with Matching Pursuit. MS Thesis, EPFL, 2000.
6. Focken, D., Stiefelhagen, R.: Towards vision-based 3D people tracking in a smart room. In *Proc. IEEE Int. Conf. on Multimodal Interfaces*, pp. 400–405, 2002.
7. Fuentes, L.M., Velastin, S.A.: People Tracking in Surveillance Applications. In *Proc. IEEE Work. on Performance Evaluation of Tracking and Surveillance*, 2001.
8. Gavrila, D. M., Davis, L. S.: 3D Model Based tracking of humans in action: a multi-view approach. In *Proc. of Computer Vision and Pattern Recognition*, pp. 73–80, 1996.
9. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. 2nd edn. Cambridge University Press, 2004.

10. Landabaso, J.L., Xu, L.Q., Pardàs, M.: Robust Tracking and Object Classification Towards Automated Video Surveillance. In *Proc. Int. Conf. on Image Analysis and Recognition*, pp. 463–470, 2004.
11. Mikic, I., Santini, S., Jain, R.: Tracking Objects in 3D using Multiple Camera Views. In *Proc. Asian Conf. on Computer Vision*, pp. 234–239, 2000.
12. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pp. 252–259, 1999.
13. Triggs, B.: Joint Feature Distributions for Image Correspondence. In *Proc. IEEE Int. Conf. on Computer Vision*, pp. 201–208, 2001.
14. Zhang, Z.: Determining the epipolar geometry and its uncertainty-a review. In *Int. Jour. of Computer Vision*, 27(2):161–195,1998.

Managing Deformable Objects in Cluster Rendering

Thomas Convard, Patrick Bourdot, and Jean-Marc Vézien

LIMSI-CNRS, Université Paris XI
{convard, bourdot, vezien}@limsi.fr

Abstract. This paper presents DRS (Distributed Rendering System), a software library for rendering dynamic 3D scenes on low cost Virtual Reality (VR) computers. This library allows the distribution of object descriptions and geometries over several graphics nodes of a PC cluster for synchronized rendering. The objects can be dynamic: in particular DRS can manage deformable objects over time. Data compression is put to profit to optimize transfer of objects geometry over the cluster nodes. This library allows real-time VR rendering on PC clusters for applications where objects are highly deformable, such as solid modeling or simulation. DRS is available as a set of C++ classes to manage the 3D objects as well as their synchronized distribution and rendering. We present experimental results in the form of examples of applications successfully ported to inexpensive distributed Virtual Reality hardware thanks to DRS.

1 Introduction

Thanks to the development of graphic accelerators in recent years, PC platforms have become powerful enough to be considered for Virtual Reality (VR) demanding applications. However, at the present time, due to some technical limitations, one single PC cannot support a full immersive environment with several stereoscopic projections such as CAVE [1] or Workbench [2]. Recent hardware solutions, such as NVidia® SLI™ or Alienware® Video Array™, allow multiple graphic boards in a single PC. But these solutions are designed to improve the rendering on a single display and are not yet validated for VR use. Therefore, at the present time, PCs have to be clustered to support VR multiple displays. PC clusters provide a low cost visualization solution, but the software development on this kind of platform is much more challenging than on high-end (thus expensive) shared memory systems commonly used for VR.

DRS is a simple library designed to distribute a graphic objects database over a PC cluster, with a special emphasis on the management of deformable objects (i.e. objects whose geometry is changing over time). Managing such objects is required to support specific applications in VR, such as 3D modeling, where the geometry of objects is ceaselessly changed by the user. The dynamic definition of objects can represent a huge amount of data that must be distributed across

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 290–297, 2005.
© Springer-Verlag Berlin Heidelberg 2005

the rendering PC nodes in order to maintain a coherent view of the VR scene on the different displays. Thus the problem is to optimize bandwidth of the network transmitting the data.

The objectives behind the design of DRS were:

1. Simplicity: DRS was designed to integrate well with other VR toolkits. Thus DRS is a software library designed to distribute rendering and perform management of 3D objects databases: it is not a VR toolkit managing devices or performing application control. Most of other scene graph distribution libraries, for instance Syzygy [3] or Avango [4] require developpers to build applications using dedicated frameworks.
2. Genericity: the aim is to support a wide range of VR applications. The primary targets of the present study were an immersive CAD demonstrator and a bioinformatics VR software previously running on a high-end visualization system.
3. Performance when dealing with deformable objects: while most available scene graphs managers handle the dynamics of scene nodes appropriately, the distribution solutions are not optimized for deformable objects.

2 Related Work

There are three main approachs in the literature for cluster graphics, each corresponding to a level of distribution: the application level, the scene graph level and the graphical commands level.

The simplest option is to replicate the VR application on each node of the cluster (Figure 2). This is the approach chosen in Net-Juggler [5], a cluster extension for VR-Juggler [6]. In this case, applications must share input events and must be synchronized at each frame buffer swap. This is particularly efficient when an application was already developped with VR-Juggler. However, this scheme is the opposite of a true distributed system: for a complex application, resource consuming tasks (surface computations, collision handling, etc.) will be replicated "as is" on each node of the cluster. In addition, for commercial applications requiring licencing, one licence must be bought for each node.

The second approach developped in Syzygy [3] or Avango [4] is to replicate only the scene descriptor (generally represented as a graph) over the PC cluster. This approach enables a single master application to manage and share synchronized updates of a scene with slave nodes using a distributed scene graph. Software proposing this solution are generally big application frameworks. Therefore, porting an existing application to such a distributed rendering environment would prove to be a tedious task.

The approach of Chromium [7] (formerly WireGL [8]), is to broadcast OpenGL commands emitted by the master to the slaves. This approach was developed in order to perform tiled rendering, but is also viable for multi-wall VR rendering systems. The main advantage of this method is that desktop applications can be ported to PC clusters without any modification of their codes, by tuning Chromium configuration files.

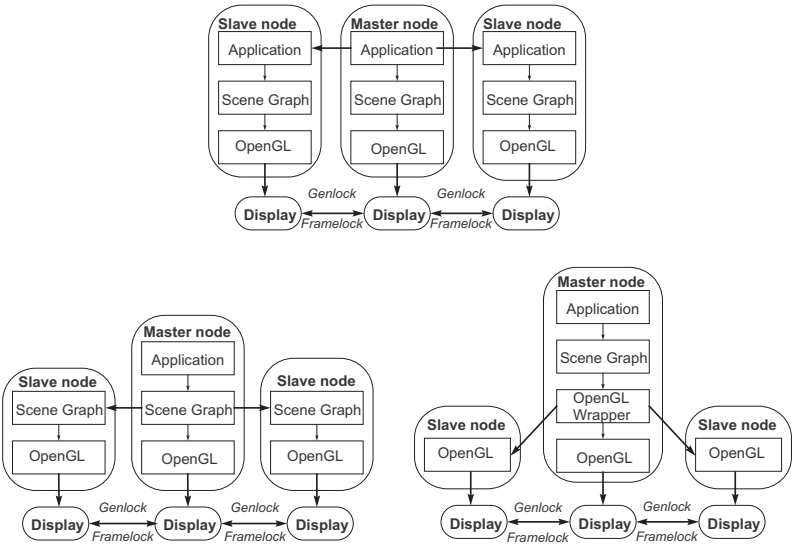


Fig. 1. (top) Application replication approach. (left) Scene graph distribution approach. (right) OpenGL commands broadcasting approach

3 DRS Approach

3.1 Architecture

DRS implements a master-slave approach (see Figure 2) by distributing VR 3D objects from the master node to the slaves nodes via networking. Each slave is a dedicated graphical node connected to a given display (e.g. a CAVE wall). VR database traversal and rendering optimizations such as frustum culling are performed on the rendering nodes of the cluster.

In addition to framelocking, the swap buffer commands on the rendering nodes must be issued within a very short time interval in order to synchronize rendering of the different nodes. DRS provides such a rendering mechanism in a transparent fashion, using sync message passing between the nodes. The master application is only in charge of telling DRS slave nodes when changes relative to the current rendered frame are to be commited.

DRS does not provide the programmer a full scene graph implementation but rather a set of classes to manage a 3D distributed database. Indeed it can be remarked that scene graphs are specific of each application field. For example CAD scene graphs are best described by a combination of CSG and B-Rep, and would prove excessively cumbersome to match CAD data to a generic scene graph format. Let us now describe in further details the DRS API.

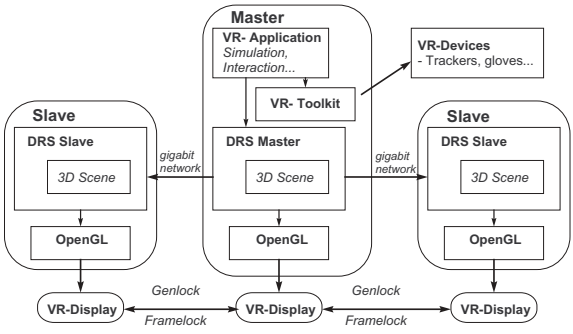


Fig. 2. DRS Architecture for a 3-PCs cluster

3.2 Objects Management

DRS maintains a database of graphical objects on both master and slaves nodes. Objects managed in the current prototype include triangular meshes, polygonal lines, points clouds, lights, materials and textures. Transformation matrices for object positions are also shared across the cluster. Projection parameters such as interocular distance or clipping planes distances can be dynamically sent by the master to the slaves in order to match the physical projection setup.

The interface between the application and DRS consists of three principal operations on each category of object:

- Create: adding a new object in the DRS database.
- Update: commiting changes on an already created object.
- Delete: suppressing an object from the database.

In the case of structured geometry, e.g. triangular meshes, three different update commands must be considered depending on what changed in the mesh (see Figure 3):

- Update topology: only the topology of the mesh has changed. In our case, the topology consists, for each triangle, in three indices corresponding to the three vertices of the triangle.
- Update geometry: only the geometry (vertices coordinates, normals or texture coordinates) of the mesh has changed. This case often occurs with triangulations performed on a regular lattice. When the surface definition changes, the coordinates of vertices in the triangulation are updated but the triangle connectivity and topology remains identical.
- Update topology and geometry: the two previous updates are performed one after the other.

3.3 Data Compression

In order to lower bandwidth requirements, mesh geometry is compressed by the master before being distributed, and the slaves must decompress it to perform

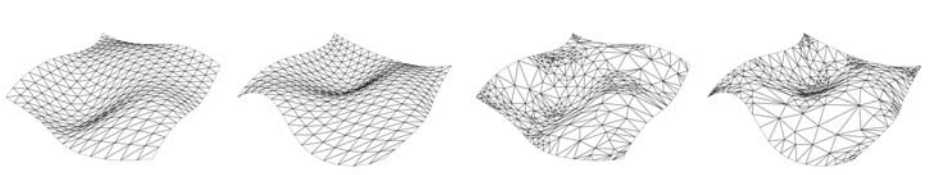


Fig. 3. Modification of NURBS surface with regular triangulation, (a) and (b), only vertices coordinates and normals need to be updated, the triangle connectivity remains the same. With adaptative triangulation, (c) and (d), both geometry and topology of the mesh change

rendering. To achieve a real optimization, the cumulated time of compressing the data, sending the compressed mesh and decompressing the data must be less than the time needed to send the uncompressed mesh. Really fast methods, without pre-computing steps, are thus needed. For example, in the case of modeling, objects are created in real-time and pre-computing complex data structure is out of the question. This constraint precludes the use of sophisticated compression algorithms, for example the ones of Taubin [9] and Rossignac [10] in which a prerequisite is the computation of the mesh connectivity graph.

To reduce the network bandwidth even further, multiresolution objects could have been envisaged. However, this kind of structure generally require costly pre-computing steps, and cannot be used in real time for distribution purposes.

Geometry compression. To compress vertex coordinates, we follow Deering's quantization approach [11]: first, vertex coordinates are normalized into a unit cube. This step requires computing the bounding box of the mesh, then performing a division per coordinate. Second, normalized coordinates are rounded off to fixed length integer. The size of each integer can be controlled, but 12 bits generally give good results for compression (36 bits per vertex instead of 96) and lead to no noticeable loss in quality for most 3D models.

Normals and texture coordinates are compressed in the same way, however the normalization step is omitted as normals components are in the $[-1, 1]$ range and texture coordinates in the $[0, 1]$ interval. They are represented on 10 bits for each component, resulting in 30 bits per normal and 20 bits per texture coordinates.

In traditional methods [11], following quantization, geometry storage is reduced with variable length encoding, generally using Huffman codes. To compress this kind of data correctly, vertex predictors are used, e.g. coding the difference between the current vertex and his ancestor in a vertex spawning tree of the mesh.

To perform well, Huffman coding requires to build a value histogram and a prefix code tree for any given dataset. Unfortunately, our experiments showed that this procedure is too time-consuming for any gain over sending raw quantized values.

Connectivity compression. A popular way to compress connectivity information is triangle strip generation [12]. A triangle strip of N triangles is represented

by $N + 2$ vertices instead of $3N$ in immediate mode. Triangle strips reduce the storage space needed and improves the rendering performance as well because they are at the hardware level with OpenGL. However, in order to build triangle strips, an adjacency graph based on the triangles of the mesh is needed, and again the computation time of such a graph is prohibitive if not readily available along with the mesh itself. In conclusion, connectivity information can be compressed fast with simple methods such as the SGI [13] algorithm, if it is computed ahead of time, as it is the case in some applications.

4 Implementation and Results

For the sake of simplicity, the DRS library was implemented over the TCP protocol. Thus it requires the master establishes one connection for each rendering slave node. To avoid this potential bottleneck the master can be equiped with several network cards : this is a solution already provided by PC cluster manufacturers.

DRS graphical nodes use pure OpenGL commands for rendering. As a consequence, DRS can be used in conjunction with any VR-toolkit that creates and manipulates an OpenGL context for drawing. For example, the projection setup for VR display on the graphical nodes can be specified with VR-Juggler [6]. DRS itself provides methods to setup CAVE-like multi-wall displays, in case users do not want to rely on an extra VR toolkit.

DRS was tested on a system configuration composed of a master PC and two slave nodes equipped with 3DLabs Wildcat graphic cards for genlocking and framelocking capabilities. The network theoretical bandwidth is 1 Gbit/s (700 Mbits/s in practice). Framelock, which is required to synchronize the frame buffer swaps of the graphics PCs, is performed via an OpenGL extension[1]. Cluster nodes can run either on Linux or on Microsoft Windows operating systems.

As expected, we noticed that the benefit of geometry compression is dependent of the speed of processors and the bandwidth of the network: CPU speed determines compression and decrompression times whereas network speed sets the data transmission time. Our program test is a simple linear elastic system deforming a mesh of about 11K vertices (25000 faces). It has been tested on a single PC, then on a networked cluster (one master + 2 slaves) with and without geometry compression. The results are conclusive (Table 1) and can be further improved with low level assembly programming (such as SSE extensions) for the compression and decompression routines.

DRS has been validated on two applications (Figure 4) ported from a high-end, shared memory, dedicated multiprocessor machine to the testing PC cluster. In a VR-CAD system, users create and edit digital mockups in an immersive environment, and the system provides real-time feedback to designers when deforming objects with detailed geometry. A DNA molecule viewer [14] was also ported to PC cluster thanks to DRS. While less demanding in terms of deforma-

[1] http://oss.sgi.com/projects/ogl-sample/registry/I3D/wgl_swap_frame_lock.txt

Table 1. DRS benchs

Configuration	Performances
Single	54 frame/s
DRS 1 Master + 2 Slave	33 frame/s
DRS 1 Master + 2 Slave with geometry compression	43 frame/s

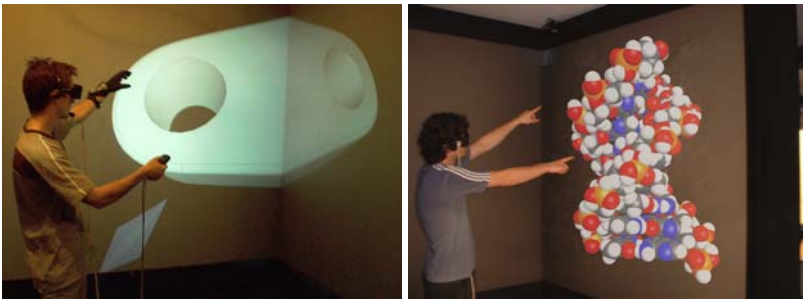


Fig. 4. Virtual Reality CAD application and DNA molecule immersive viewer in a CAVE-like system running on a PC cluster with DRS

tion, this later application typically manipulates thousands of DNA bases while displaying entire chromosomes.

5 Conclusion

In this paper, a 3D database distribution library for VR rendering on a PC cluster was presented. DRS provides a simple yet comprehensive interface to programmers wishing to port their application to VR immersive displays, while retaining high performance regarding the distribution of deformable objects. DRS was successfully put to use by allowing two existing applications (a CAD modeling experimental software and a DNA molecule viewer) to be successfully ported from a high-end visualization system to a PC cluster.

DRS could be extended to managing other modalities, e.g. haptics. Indeed a similar 3D objects distribution problems arise in haptics when connecting more than one force feedback or vibrotactile device to the VR system. In association with a haptic rendering software, DRS can allow multiple haptic systems to share a synchronized version of a dynamic 3D database. Because haptics need high refresh rates and detailed geometry description, DRS will probably be especially suited to force-feedback distributed rendering applications.

A natural evolution of DRS would consist in using multicast protocol to lower the necessary bandwidth. Multicast would allow the master cluster node to send data only once to all the slaves. However, the drawbacks of multicast are packet loss and packet mis-ordering. To manage these problems, checking algorithms have to make sure the data received by the slaves are correct. These techniques induce significant overhead and reduce the available bandwidth. Several prelim-

inary tests on our system (consisting in 2 slaves and one master), showed that overhead is too important compared to the actual multicast gain. However in a 6 walls/6 slaves configuration, multicasting can be reconsidered.

For the moment, DRS can deal with triangular meshes, polylines and points clouds. The system could be improved by supporting NURBS curves and surfaces or conics for example. These primitives would allow to reduce the data flow for specific applications, such as CAD for the NURBS and molecular visualization for the conics.

References

1. Cruz-Neira, C., Sandin, D., DeFanti, T.: Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In: Computer Graphics (Proceedings of SIGGRAPH '93). (1993) 135–142
2. Kruger, W., Bohn, C., Frohlich, B., Schuth, H., Strauss, W., Wesche, G.: The Responsive Workbench: A Virtual Work Environment. IEEE Computer **28** (1995) 42–48
3. Schaeffer, B., Goudeseune, C.: Syzygy: Native PC Cluster VR. In: Virtual Reality Conference '03, IEEE (2003) 15–22
4. Tramberend, H.: Avango: A Distributed Virtual Reality Framework. In: Proceedings of Afrigraph '01. (2001)
5. Allard, J., Gouranton, V., Lecointre, L., Melin, E., Raffin, B.: Net Juggler : Running VR Juggler with Multiple Displays on a Commodity Component Cluster. In: Virtual Reality Conference '02, Orlando, Florida, IEEE (2002) 273
6. Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., Cruz-Neira, C.: VR Juggler: A Virtual Platform for Virtual Reality Application Development. In: Virtual Reality Conference '01, Yokohama, Japan (2001)
7. Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P.D., Klosowski, J.T.: Chromium: A Stream Processing Framework for Interactive Rendering on Clusters. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002) **21** (2002) 693–702
8. Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M., Hanrahan, P.: WireGL: A Scalable Graphics System for Clusters. In: Proceedings of SIGGRAPH 2001. (2001) 129–140
9. Taubin, G., Rossignac, J.: Geometric Compression Through Topological Surgery. ACM Transactions on Graphics **17** (1998) 84–115
10. Rossignac, J.: Edgebreaker: Connectivity Compression for Triangle Meshes. IEEE Transactions on Visualization and Computer Graphics **5** (1999) 47–61
11. Deering, M.: Geometry Compression. In: Computer Graphics (Proceedings of SIGGRAPH '95). (1995) 13–20
12. Evans, F., Skiena, S., Varshney, A.: Optimizing Triangle Strips for Fast Rendering. In: Proceedings IEEE Visualization '96. (1996) 319–326
13. Akeley, K., Haeberli, P., Burns, D.: tomesh.c. C Program on SGI Developer's Toolbox CD (1990)
14. Hérisson, J., Gros, P.E., Férey, N., Magneau, O., Gherbi, R.: DNA in Virtuo: Visualization and Exploration of 3D Genomic Structures. In: Proceedings Afrigraph '04. (2004) 35–40

Revolute Quadric Decomposition of Canal Surfaces and Its Applications

Jinyuan Jia[1], Ajay Joneja[2], and Kai Tang[2]

[1] Zhuhai College of Jilin University,
Jinwan District, Zhuhai, P. R. China, 113001
csjyjia@yahoo.com.cn
[2] The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong, P. R. China
{Joneja, mektang}@ust.hk

Abstract. Surfaces subdivision is an important means for geometric computing of surfaces in CAD. This paper proposes a new quadric subdivision for canal surfaces in this paper, *RQ-sphere decomposition*, that subdivides canal surfaces as a set of truncated revolute quadric with joint spheres. Experimental results show that the RQ-sphere decomposition is better than existing methods.

1 Introduction

Canal surfaces are very useful for representing long thin objects, for instance, poles, 3D fonts, brass instrument or internal organs of the body in solid modeling and CG/CAD/CAM. It includes natural quadrics (cylinder, cone and sphere), revolute quadrics, tori, pipes and Dupin cyclide. Therefore, some algorithms of geometric computing around canal surfaces have been developed recently [2], [3], [6], [8], [9], [10], [11], [12] and [13], in which, the decomposition schemes of canal surfaces play an important role. Although quadrilateral/triangular subdivision is a general scheme to subdivide a general surface into a quadrilateral/triangular mesh, however, it is too general so that it becomes inefficient when applied to canal surfaces directly. Canal surfaces possess some good geometric properties and should have more efficient decomposition scheme than triangulation. This paper presents new revolute quadric decomposition scheme for canal surfaces.

Besides quadrilateral or triangular decomposition, the three decomposition schemes have been given specially for canal surfaces: (1) circle decomposition, (2) cone-sphere decomposition and (3) cyclide decomposition. *Circle decomposition* subdivides a canal surface as a densely set of characteristic circles along its spine curve. It reduces some geometric computations of canal surfaces to the corresponding ones on circles. Several geometric problems of canal surfaces have been given in [5], [6], [8] and [10] based on circle decomposition. *Cone-sphere* [11] samples a set of spheres along its spine curve and joint each two adjacent sampling spheres with a common tangential truncated cone, thus, a canal surface can be approximated with a set of alternating truncated cones and spherical strips. However, it has two drawbacks: (1) at the high precision, a very dense set of cone-sphere pairs

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 298 – 305, 2005.
© Springer-Verlag Berlin Heidelberg 2005

are required, and (2) when the shape of canal surface bends at a big curvature or torture, two neighboring truncated cones may intersect each other, see Fig 3(a), that is called self-intersection. *Cyclide decomposition* [13] subdivides a canal surface as a group of bounded Dupin cyclide pairs. This is a natural decomposition scheme for canal surfaces. However, cyclide is a degree-four surface, its related geometric computing have no closed form solutions, thus, it does not reduce geometric computing complexity of canal surfaces.

Quadrics are the lowest order curved surfaces. They have good parametric and implicit forms. Some related geometric computations are relatively easy and simple. Therefore, how to construct a piecewise quadric net of a surface becomes important in CAGD and some methods have been proposed on constructing a piecewise quadric spline to approximate an arbitrary surface, see [1], [4], [14] and [15]. However, while suitable for general surfaces, the aforementioned schemes become inefficient or too complicated when they are directly applied to special surfaces. It is natural to explore the specific quadric decomposition schemes for canal surface. It has some useful geometric properties that can just be exploited to develop more efficient quadric decomposition schemes.

In this paper, we propose a revolute quadric-spherical stripes decomposition for a regular canal surfaces, which can avoid the two problems of the cone-sphere scheme above. The main idea is (1) to sample a set of spheres along the spine curve appropriately and (2) construct a common tangential revolute quadric between each pair of adjacent sampling spheres, instead of the common tangential truncated cone, to approximate the canal surface. We note that the revolute quadric-spherical sphere (hereafter RQ-sphere) scheme is a natural extension of cone-sphere decomposition scheme.

2 Preliminary

We assume that a canal surface with a regular spine curve $\mathbf{C}(t)$ and a rational radius function $R(t)$ is valid, $||\mathbf{C}'(t)|| \neq 0$, and $||\mathbf{C}'(t)||^2 - R(t)^2 > 0$. $\mathbf{C}_k(t)$ (see Fig. 1) and $R_k(t)$ are the center point and the radius of its characteristic circle. Then its equation $K(t, \theta)$ can be given as the following *Frenet* form ($\mathbf{T}(t)$, $\mathbf{N}(t)$, $\mathbf{B}(t)$) of $\mathbf{C}(t)$, see [8] and [9].

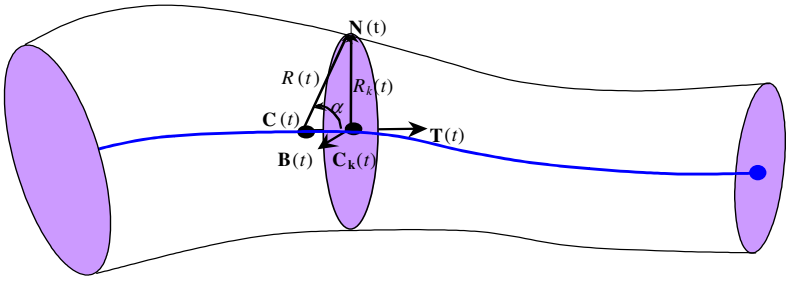


Fig. 1. The characteristic circle of canal surfaces

$$\mathbf{K}(t,\theta) = \mathbf{C_k}(t) + R_k(t)(\cos(\theta)\mathbf{N}(t) + \sin(\theta)\mathbf{B}(t)). \quad (1)$$

where

$$\mathbf{T}(t) = (x'(t), y'(t), 0), \qquad \mathbf{B}(t) = (0, 0, 1),$$

$$\mathbf{N}(t) = \frac{(-y'(t), x'(t), 0)}{\sqrt{x'(t)^2 + y'(t)^2}}, \quad \mathbf{C_k}(t) = \mathbf{C}(t) - R(t)R'(t)\frac{\mathbf{C}'(t)}{\|\mathbf{C}'(t)\|^2}, \quad R_k(t) = R(t)\frac{\sqrt{\|\mathbf{C}'(t)\|^2 - R'(t)^2}}{\|\mathbf{C}'(t)\|}$$

3 RQ-Sphere Decomposition of Canal Surfaces

In this section, we address three problems: (*a*) how to construct a simple RQ-sphere fitting a canal surface for two given sampling spheres on the spine curve, (*b*) how to avoid self-intersection of two neighboring RQs, and (*c*) how to optimize local fitting between a RQ-Sphere and the canal surface.

3.1 Construction of Local RQ-Sphere Fitting

We construct a local coordinate system with the centers of two sampling spheres S_1 and S_2 at (0, 0, 0) and (c, 0, 0) respectively as shown in (*a*) of Fig. 2, and the revolute axis of the common tangential revolute quadric of S_1 and S_2 as x-axis. Then

Sphere S_1: $x^2 + y^2 = r^2$, (2)

Sphere S_2: $(x-c)^2 + y^2 = r^2$, (3)

Revolute quadric RQ: $Ax^2 + y^2 + Dx + F = 0$. (4)

By substituting Eq. (2) into Eq. (4), we get a quadratic equation $(A\text{-}1)x^2 + Dx + (F+r^2) = 0$, since the RQ is tangential to S_1, its determinant must be zero:

$$\Delta_1 = D^2 - 4(A-1)(F+r^2) = 0. \quad (5)$$

Similarly by substituting Eq. (3) into Eq. (4), we get equation $(A\text{-}1)x^2 + (D+2c)x + (F + R^2 - c^2) = 0$, its determinant also should be zero, since RQ is tangential to S_2:

$$\Delta_2 = (D+2c)^2 - 4(A-1)(F+R^2-c^2) = 0. \quad (6)$$

From Eq. (5) and Eq. (6), we can obtain the other two coefficients D and F as functions of A:

$$D = \frac{A(R^2 - r^2 - c^2) + (r^2 - R^2)}{c}, \qquad F = \frac{D^2}{4(A-1)} - r^2. \quad (7)$$

As shown in Fig. 2, there are totally four cases of the shape of RQ: common tangential hyperboloid, ellipsoid, paraboloid and truncated cone, depending on the value of the coefficient A. Let us analyze the impact of A on the shape of common tangential revolute quadric.

1. If $A = 0$, Eq. (4) is reduced to $y^2 + Dx + F = 0$;
2. If $A \neq 0$, Eq. (4) can be transformed into

$$A\left(x + \frac{D}{2A}\right)^2 + y^2 = G \tag{8}$$

where

$$G = \frac{D^2}{4A} - F = \frac{D^2}{4A} - \frac{D^2}{4(A-1)} + r^2 = D^2 \frac{-1}{4A(A-1)} + r^2 \tag{9}$$

The x-coordinates of the tangent points on the two end spheres S_1 and S_2 can be written as:

$$T_{x1} = \frac{-D}{2(A-1)}, \quad T_{x2} = \frac{-(D+2c)}{2(A-1)}, \quad T_{x2} - T_{x1} = \frac{c}{1-A} \tag{10}$$

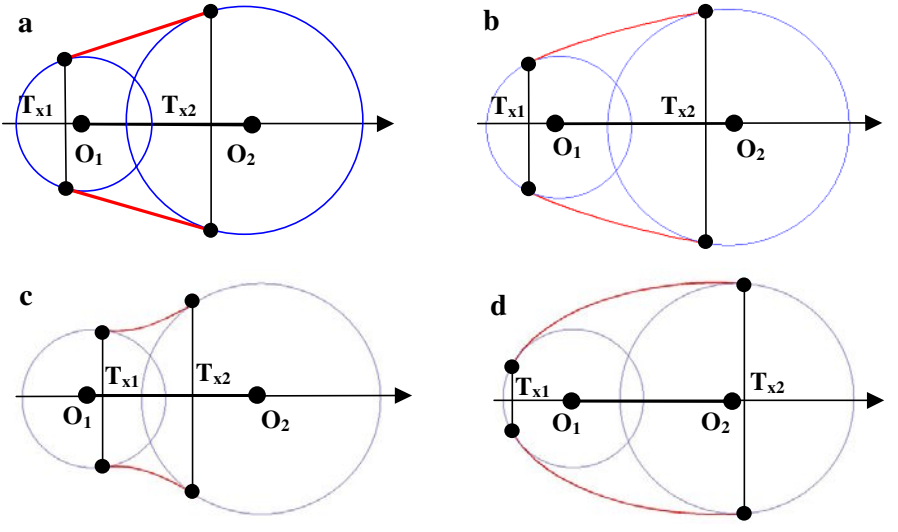


Fig. 2 Four types of RQ-Sphere fitting scheme**:** ***a.*** Cone-sphere fitting, ***b.*** Paraboloid-sphere fitting, ***c.*** Hyperboloid-sphere fitting, ***d.*** Ellipsoid-cone fitting

Practically, we require that $T_{x1} \leq T_{x2}$; thus implies $A < 1$, since $c > 0$. Furthermore, we can classify the scope of A as follows:

1. If $0 < A < 1$, $G > 0$, the corresponding RQ of Eq. (4) is an ellipsoid;
2. If $G = 0$, Eq. (4) represents a truncated cone. So cone-sphere is a special case of RQ-sphere decomposition;
3. If $A < 0$ and $G > 0$, Eq. (4) represents a hyperboloid of one leaf.
4. If $A < 0$ and $G < 0$, Eq. (4) represents a hyperboloid of two leaves.

3.2 Overlapping Avoidance of Two Adjacent RQs

When the curvature of the spine curve changes acutely in cone-sphere fitting, two adjacent fitting truncated cones TC_1 and TC_2 for three specified sampling spheres may intersect each other as shown in Fig. 3, it is noted as self-intersection. However, two fitting revolute quadrics can avoid such self-intersections by adjusting the coefficient A of two fitting RQs appropriately. That introduces two questions: (1) how to judge such self-intersection of two fitting RQs; and (2) how to the best A for optimal RQ-sphere fitting to canal surface. To the first question, we give a theorem, the necessary and sufficient condition for self-intersection determination. To the second question, we will address it in the nest subsection.

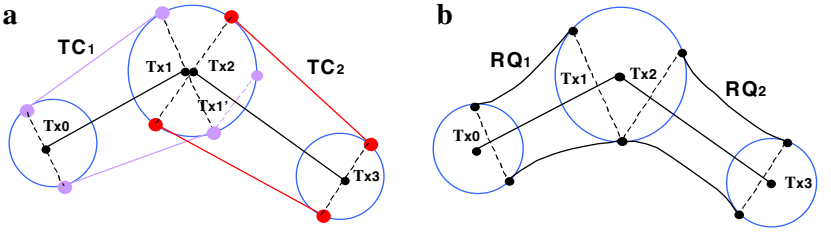


Fig. 3. Avoiding self-intersection of RQ-Sphere fitting, ***a.*** Self-intersection of cone-sphere fitting, ***b.*** No self-intersection of RQ-Sphere fitting

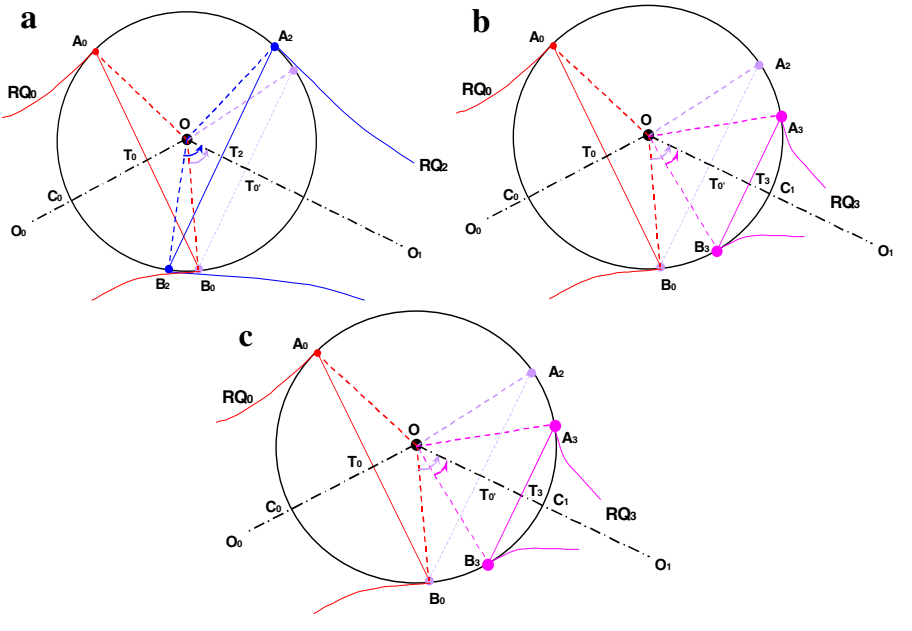


Fig. 4. Three cases of two adjacent fitting RQs: *a.* Tangential case of two adjacent RQs, *b.* Overlapping case of two adjacent RQs, *c*. Separated case of two adjacent RQs

Algebraically, determining intersections of two RQs requires solution of quartic equation in general. Here, we give a geometric method to solve it. The overlapping of two RQs and their coherent sphere O in 3D are equivalent to the corresponding problems of the chords A_0B_0 and A_1B_1 (A_2B_2 or A_2B_2, as shown in (*a*), (*b*) and (*c*) of Fig. 4 respectively) in 2D. That concludes the following theorem.

Theorem 1. Two adjacent revolute quadrics RQ_1 and RQ_2 have the common joint sphere, they overlap each other if and only if two corresponding chords A_0B_0 and A_1B_1 intersect each other, furthermore, if and only if $\|OT_0\| > \|OT_1\|$ holds.

It is much simpler than algebraic method. The proof is trivial and omitted here due to limited space.

3.3 Locally Optimal RQ-Sphere Fitting

By sampling a canal surface $x = x(u,v)$, $y = y(u,v)$ and $z = z(u,v)$ with ($u_i = u_0, u_1, \ldots\ldots, u_m$ and $v_i = v_0, v_1, \ldots\ldots, v_n$, the least square fitting method is employed to determine the best value for A in the sense of minimizing the overall sum of square distances of the sampling points (x_{ij}, y_{ij}, z_{ij}) to the fitting RQ-sphere pairs, which can be represented as:

$$\min_{A,D,F} f(A,D,F) = \sum_{i,j} (Ax_{ij}^2 + y_{ij}^2 + z_{ij}^2 + Dx_{ij} + F)^2 \tag{11}$$

By differentiating (27) with regards to A, we get

$$\min\ f_A'(A,D,F) = 2\sum_{i,j}^{N} (Ax_{ij}^2 + y_{ij}^2 + z_{ij}^2 + Dx_{ij} + F)(x_{ij}^2 + D_A' x_{ij} + F_A') \tag{12}$$

By substituting the following formulae:

$$D = mA + n,\ D_A' = m,\ F = \frac{(mA+n)^2}{4(A-1)} - r^2,\ F_A' = \frac{2(mA+n)m(A-1) - (mA+n)^2}{4(A-1)^2} = k_0 + \frac{k_2}{4(A-1)^2}, \tag{13}$$

where $k_0 = m^2/4$ and $k_2 = (m - n)^2$, into (11), we obtain an optimization equation about A as follows:

$$\begin{aligned} f'(A) &= 2\sum_{i,j}^{N} \left(Ax_{ij}^2 + y_{ij}^2 + z_{ij}^2 + (mA+n)x_{ij} + \frac{(mA+n)^2}{4(A-1)} - r^2 \right) \left(x_{ij}^2 + mx_{ij} + k_0 + \frac{k_2}{4(A-1)^2} \right) \\ &= M_0 A(A-1)^3 + M_1(A-1)^3 + M_2(mA+n)(A-1)^3 + M_0(mA+n)^2(A-1) + \\ &\quad + M_3 k_2 A(A-1) + M_4 k_2 A(A-1) + M_5(mA+n)k_2 + (mA+n)^2 k_2 = 0 \end{aligned} \tag{14}$$

where

$$M_0 = \frac{16}{N}\sum_{i,j}^{N} x_{ij}^2 (x_{ij}^2 + mx_{ij} + k_0),\quad M_1 = \frac{16}{N}\sum_{i,j}^{N} (y_{ij}^2 + z_{ij}^2 - r^2)(x_{ij}^2 + mx_{ij} + k_0),\quad M_2 = \frac{16}{N}\sum_{i,j}^{N} x_{ij}(x_{ij}^2 + mx_{ij} + k_0),$$

$$M_3 = \frac{4}{N}\sum_{i,j}^{N} x_{ij}^2,\quad M_4 = \frac{4}{N}\sum_{i,j}^{N} (x_{ij}^2 + mx_{ij} + k_0),\quad M_5 = \frac{4}{N}\sum_{i,j}^{N} x_{ij}$$

Actually, Eq. (13) is a quartic equation and it has closed form solutions to A, D and F for a locally optimal RQ-sphere fitting canal surfaces.

4 Illustrative Examples

We implemented both cone-sphere and RQ-sphere decomposition of canal surfaces for comparison. Three examples are illustrated in Fig. 5, the spine curve and radius of them are represented by cubic Bezier curve in Examp2 and Examp3. The performance data in Table 1 shows that, for the same canal surface, the ratio of the required RQ-sphere pairs *vs.* cone-sphere pairs nearly approaches 1 : 2. We also found that when the tolerance becomes very tiny value or the curvature of the spine curve big enough, cone-sphere results in local self-intersection unavoidably. However, RQ-sphere can avoid such local self-intersection very well.

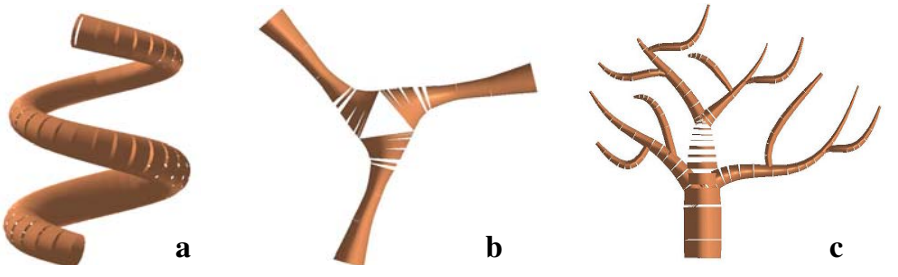


Fig. 5. Applications of RQ-sphere decompositions to canal surfaces

Table 1. Performance data of cone-sphere vs. RQ-sphere for fitting canal surfaces

# of Quadric-Sphere Pairs	Cone-sphere	RQ-sphere
Example 1 at 0.5×10^{-5}	235	175
Example 2 at 0.5×10^{-4}	1373	636
Example 3 at 0.5×10^{-3}	2123	768

5 Conclusions

Compared to the cone-sphere scheme, (1) RQ-Sphere can fit canal surfaces more flexibly to avoid self-intersection and yield visually smother rendering; (2) it has one degree-of-freedom to allow optimal approximation of canal surfaces, and (3) it only requires fewer fitting components at the same approximation quality. The revolute quadric decomposition for surfaces of revolution by authors previously in [7] is a special case of RQ-sphere decomposition for canal surfaces. The following works can be extended along this paper:

1. Reconstruction of boundary canal surfaces from skeleton curves by using MAT technique becomes important. The reconstruction process in [2] and [3] (for one branch of the MAT) is exactly the definition of canal surface, (the medial axis is the spine curve C(t), the resulting canal surface is the boundary of reconstructed volume) Practically, the medial axis is given by a sequence of discrete sampling

points. In fact, the process of Gelston and Dutta's reconstruction is cone-sphere fitting. Thus, our RQ-sphere can be extended to such reconstruction directly.

2. RQ-sphere decomposition can reduce the intersection computing problems of canal surfaces (ray/canal, plane/canal and canal/canal) to the corresponding problems of RQs. Also, other geometric computing problems of canal, i.e. isophote, silhouette, and so on, can be revisited in the new perspective of RQ-sphere decomposition.

Although RQ-sphere decomposition is better than cone-sphere decomposition, it still requires rather huge number of RQ-sphere pairs to approximate canal surfaces at the high precision, so more efficient quadric decomposition for canal surfaces and even other surfaces is our existent goal in the long run.

References

1. Bangert, C. and Prautzsch, H.: Quadric Spline. In: CAGD, Vol. 16, (1999) 497-515.
2. Ferly, E., Gascuel C. M-P, Attali D.: Skeletal Reconstruction of Branching Shapes. In: Implicit Surfaces'96, Eindhoven, Netherlands, October, (1996) 127-142.
3. Gelston S. M. and Dutta D.: Boundary Surface Recovery from Skeleton Curves and Surfaces". In: CAGD, Vol. 12, (1995) 27-51.
4. Guo B.: Quadric and Cubic Bitetrahedral Patches". In: The Visual Computer, Vol. 7, (1995) 253-262.
5. Heo H. S., Hong S. J., Seong J. K. and Kim M. S.: The Intersection of Two Ringed Surfaces and Some Related Problems". In: Graphical Model, Vol. 63, (2001) 228-244.
6. Johnstone J.: A New Intersection Algorithm for Cyclides and Swept Surfaces Using Circle Decomposition. In: CAGD, Vol. 10, (1993) 1-24.
7. Jia J. Y., Baciu G. and Kwok K. W.: Quadric Decomposition for Computing The Intersections of Surfaces of Revolution. In Graphical Models, Vol. 55, (2004) 363-383.
8. Kim K.-J. and Lee I.-K.: The Perspective Silhouette of A Canal Surface. In: Computer Graphics Forum, Vol. 22, (2003) 15-22.
9. Kim K.-J. and Lee I.-K.: Computing Isophotes of Surface of Revolution and Canal Surface. In: CAD, Vol. 5, (2003) 215-223.
10. Kim K. J., "Minimum Distance Between A Canal Surface and A Simple Surface. In CAD, Vol. 35, (2003) 871-879.
11. Nelson Max: Cone-Spheres. In ACM SIGGRAPH Computer Graphics, Vol. 24, (1991) 59-62.
12. Nishita T. and Hohan H.: A Scan Line Algorithm for Rendering Curved Tubular Objects. In: Proceedings of Pacific Graphics, (1999) 92-101.
13. Paluszny, M. and Bühler, K.: Canal Surfaces and Inversive Geometry. In: Mathematical Methods for Curves and Surfaces II, (1998) 367-375.
14. Powell, M. J. D. and M. A. Sabin: Piecewise Quadratic Approximations on Triangles. In: ACM Transaction on Mathematical Software, Vol. 3, (1977) 316-325.
15. Sed[1]erberg, T.: Piecewise Algebraic Surface Patches. In. CAGD, Vol. 2, (1985) 53-60.

[1] This work was partially supported by Liaoning Educational Science and Technology Fund.

Adaptive Surface Modeling Using a Quadtree of Quadratic Finite Elements

G. P. Nikishkov

University of Aizu, Aizu-Wakamatsu 965-8580, Japan
niki@u-aizu.ac.jp
http://www.u-aizu.ac.jp/∼niki

Abstract. This article presents special quadrilateral quadratic refinement elements, which provide geometry and field continuity across T-junctions where two elements are connected to one side of a larger quadrilateral. The main idea in element refinement is to place one or more nodes outside the element area and to modify element shape functions in order to maintain continuity at refinement edges. Special refinement elements allow one to adaptively refine a mesh in such a way that it fits a quadtree data structure. An algorithm of surface modeling starts with a coarse mesh of quadratic quadrilateral elements. Adaptive mesh refinement is done in an iterative manner. At each iteration, the finite element equation system is solved to provide nodal locations with minimization of global approximation error. Elements with excessive local errors are split into four new elements. The mesh refinement iteration process is terminated when no element splits occur. The created mesh of quadratic quadrilaterals can be used directly in finite element analysis.

1 Introduction

Surface modeling is important for many fields of activity ranged from industrial CAD to animation in computer art. This paper is focussed on surface modeling for engineering applications, which use finite element analysis. The basic objective of this type of surface modeling can be stated as follows: generate a surface mesh that approximates a surface with specified accuracy and that can be used for finite element analysis.

Specified accuracy of surface approximation can be achieved with the use of adaptive algorithms. Adaptive mesh optimization algorithms for surface approximation have been considered in [1, 2, 3]. Duchineau et al. [1] presented an adaptive algorithm for constructing triangular meshes with guaranteed error bounds. Parajola [2] constructed restricted quadtrees for storing blocks of triangles in quadtree leafs. Minimization of global approximation error was done by Grosso et al. [3] using solution of a variational problem. However, the authors of these publications employed triangular elements. While triangular elements can be used in the finite element analysis, quadrilateral elements are preferable because of their better properties for solution of variational problems [4]. It is

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 306–313, 2005.
© Springer-Verlag Berlin Heidelberg 2005

known that quadrilaterals with quadratic interpolating functions are more efficient than linear quadrilateral elements. Generation of a topologically regular quadrilateral mesh by a mapping technique is relatively easy [5]. However, performing local mesh refinement for a quadrilateral mesh is considered a difficult task [6]. For example, a subdivision scheme for quadrilateral meshes proposed by Kobbelt [7] produces regular meshes with some singular vertices.

A natural process of local refinement for quadrilateral meshes is subdivision of one quadrilateral element into four smaller quadrilaterals. However, so-called "T-junctions" where two elements are connected to one side of a larger element are created after such subdivisions. Such meshes cannot be employed in visualization and modeling because of geometrical and functional discontinuity across T-junctions. Forting and Tanguy [8] considered connection of two Lagrangian 9-node elements to one side of the same type larger element. Displacement compatibility was enforced by introduction of additional constraints in the global finite element equation system. This provides convergence of the finite element solution but the finite element mesh remains geometrically incompatible. Sederberg et al. [9] dealt with T-junction meshes using a generalization of B-splines, which they call T-splines.

The main contribution of this article is an introduction of special quadrilateral refinement elements, which provide continuity across element T-junctions. In refinement elements, one or more midside nodes are placed outside the element area and element shape functions are modified in order to maintain continuity at refinement edges. Shape functions for refinement elements are derived in Section 2. In Section 3, a finite element procedure for surface modeling with global error minimization is presented. Section 4 describes an iterative algorithm for adaptive surface approximation, which produces a quadtree of quadratic quadrilaterals with guaranteed local approximation error. In Section 5, an example of surface modeling is given.

2 Refinement Quadrilaterals

An isoparametric quadratic quadrilateral element [4] with eight nodes is depicted in Fig. 1,a. Any field (including coordinates) specified at element nodes is interpolated inside the element with the use of shape functions N_i, which are defined in local coordinates $\xi,\ \eta\ (-1 \le \xi,\ \eta \le 1)$:

$$\begin{aligned} u(\xi,\eta) &= \sum N_i(\xi,\eta)u_i\ , \\ N_i &= \frac{1}{2}(1-\xi^2)(1+\eta\eta_i)\ ,\quad i=2,\ 6\ , \\ N_i &= \frac{1}{2}(1+\xi\xi_i)(1-\eta^2)\ ,\quad i=4,\ 8\ , \\ N_i &= \frac{1}{4}(1+\xi\xi_i)(1+\eta\eta_i)-\frac{1}{2}(N_{i-1}+N_{i+1})\quad i=1,\ 3,\ 5,\ 7\ , \end{aligned} \tag{1}$$

where $u(\xi,\eta)$ is a field value at some point inside the element; u_i are field values at element nodes; ξ_i and η_i are values of the local coordinates ξ and η at the node i.

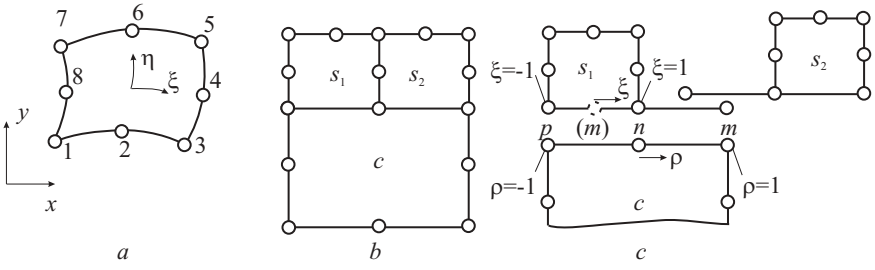


Fig. 1. a) Quadratic isoparametric quadrilateral element; b) Mesh refinement 1:2; c) Configuration of special elements

Using ordinary quadrilateral elements it is difficult to make mesh refinement. We present here special refinement elements that allow T-junctions as shown in Fig. 1,b. In special refinement elements, some midside nodes are moved outside the element area in order to provide geometry and field continuity. An idea of mesh refinement with special elements is illustrated in Fig. 1,c. Mesh T-junction is created using two refinement elements s_1 and s_2, which have nodes located outside of the element area.

Shape functions in the refinement element should provide continuity across T-junction. In order to have continuity along the element edge $p-n-m$ (Fig. 1,c) the field value at point $\xi = 0$ in the special element s_1 should be equal to the field value at point $\rho = -0.5$ in the conventional neighboring element c: $u(\xi = 0) = u(\rho = -0.5)$. Using interpolation (1), a compatibility equation can be written as:

$$u(\xi = 0) = u(\rho = -0.5) = \frac{3}{8}u_p + \frac{3}{4}u_n - \frac{1}{8}u_m \,. \tag{2}$$

It is possible to demonstrate that the compatibility equation can be satisfied if the shape functions N_i in the refinement element are modified as follows:

$$N'_p = N_p + \frac{3}{8}N_m \,, \quad N'_m = -\frac{1}{8}N_m \,, \quad N'_n = N_n + \frac{3}{4}N_m \,. \tag{3}$$

Interpolation of a field, which is equal to 1 just at one corner node of the conventional element, is shown in Fig. 2,a. Same field interpolation in the refinement element is depicted in Fig. 2,b. Refinement elements preserve quadratic interpolation for both geometry and field modeling.

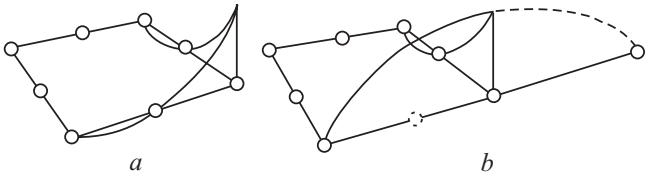


Fig. 2. Corner shape function in an ordinary quadratic quadrilateral (a) and in a refinement element

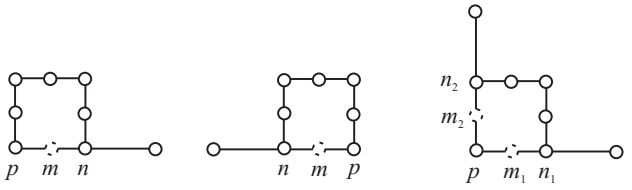


Fig. 3. Typical configurations of special refinement elements

Configurations of special refinement elements useful for quadtree mesh subdivision are shown in Fig. 3. If a special element contains two outside nodes as shown in Fig. 3 on the right, then modification (3) is applied twice using nodes m_1 and m_2:

$$\begin{aligned} &N'_p = N_p + \frac{3}{8}(N_{m_1} + N_{m_2})\,, \quad N'_{m_1} = -\frac{1}{8}N_{m_1}\,, \quad N'_{n_1} = N_{n_1} + \frac{3}{4}N_{m_1}\,, \\ &N'_{m_2} = -\frac{1}{8}N_{m_2}\,, \quad N'_{n_2} = N_{n_2} + \frac{3}{4}N_{m_2}\,. \end{aligned} \tag{4}$$

3 Global Error Minimization

Global error minimization of surface approximation can be done with the use of the variational approach to functional minimization. It is assumed that a surface that should be approximated is defined explicitly or by a point cloud. An element approximation error E_e for a surface segment covered by element e can be defined as:

$$E_e = \int_{A_e} (f-u)^2 dA = \int_{A_e} (f - \sum N_i u_i)^2 dA\,, \tag{5}$$

where f is the specified surface height; u is the height approximated by the eth element; u_i are nodal values of u; N_i are the shape functions and A_e is the element area. Element error can serve as a local error indicator for mesh refinement. Global approximation error is the sum of element errors:

$$E = \int_A (f-u)^2 dA = \sum_e E_e = \sum_e \int_{A_e} (f - \sum N_i u_i)^2 dA\,. \tag{6}$$

Minimization of the global error functional E results in the following global finite element equation system:

$$\mathbf{KU} = \mathbf{R}, \quad k^e_{ij} = \int_{A_e} N_i N_j dA, \quad r^e_i = \int_{A_e} N_i f dA\,, \tag{7}$$

where $\mathbf{K}$ is a global "stiffness" matrix; $\mathbf{U}$ is a vector of nodal height values and $\mathbf{R}$ is a right-hand side vector. The global matrix $\mathbf{K}$ and the right-hand side vector $\mathbf{R}$ are composed of element "stiffness" matrices $\mathbf{k}_e$ and element vectors $\mathbf{r}_e$ using element connectivity information.

Iterative methods can be efficiently used for solution of the finite element equation system (7) since good initial approximation for the solution vector **U** is available in the beginning of each refinement step.

4 Surface Modeling Algorithm

Surface modeling starts with a coarse mesh consisting of usual quadrilaterals. If the approximation domain is a quadrilateral area then it is possible to employ just one quadratic quadrilateral element in the beginning. According to the adaptive mesh refinement procedure, refinements are carried out for elements where an error indicator (5) is greater than a specified error tolerance.

The quadrilateral element that should be refined, is replaced by four quadrilaterals. Some or all of these new quadrilaterals are special refinement elements. Examples of one element refinement for the case of regular element surrounding and for the case of a partially refined mesh are shown in Fig. 4

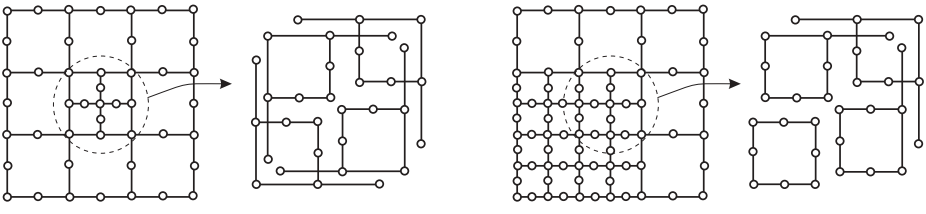


Fig. 4. Examples of 1:4 element refinement for the case of regular element surrounding and for the case of a partially refined mesh

Mesh data after any number of 1:4 element refinements is stored in a quadtree data structure. If to adopt that a quadtree root can be depicted as a square then each quadtree node corresponds to a smaller square. Internal nodes have exactly four children. Leafs (nodes without children) represent actual elements, which can be usual elements or special refinement elements. After splitting elements with excessive errors, the refined mesh can contain places where one element have a connection to more than two elements. This makes impossible to provide continuity across element boundaries. A quadtree where large squares can be adjacent to more than two smaller squares at any edge is called unbalanced [10]. Quadtree balancing is performed by additional element subdivisions. For any balanced quadtree, the geometry and field continuity can be ensured by using refinement elements with outside nodes.

An adaptive algorithm of surface modeling is organized as an iterative procedure. During each iteration, first, the finite element equation system (7) is assembled and solved. The height surface values are obtained for all mesh nodes. Then local element errors E_e (5) are estimated and 1:4 splits of elements with excessive errors are performed. The element quadtree is balanced at the end of each iteration. Mesh refinement process is terminated if no splits occur during

current iteration. Pseudo code of the surface modeling algorithm is presented below.

```
Initialize Quadtree Q
do
   Assemble and solve equation (7)
   Number of splits s= 0
   for Element e=1 to Number of Elements
      Compute approximation error E_e (5) for Element e
      if E_e > Error Tolerance then
         s = s + 1
         Split Element e into four elements
         Put new elements into Quadtree Q
      end if
   end for
   if s > 0 balance Quadtree Q
while s > 0  .
```

The algorithm of surface modeling has been implemented in Java. Surface is visualized using Java 3D.

5 Example

The proposed algorithm of adaptive surface modeling with a quadtree of quadratic quadrilaterals is demonstrated on the approximation of the following surface defined on a square domain:

$$f = 0.5e^{-0.16(x^2+y^2)} \sin(2x)\cos(2y), \qquad -5 \le x \le 5, \quad -5 \le y \le 5 \; .$$

The height range for the above height function is $[-0.5, 0.5]$ and the size in height direction is 1. The following error measure is used for mesh refinement:

$$\bar{E}_e = \sqrt{\frac{1}{A_e}\int_{A_e} (f-u)^2 dA} \; ,$$

where f is the specified surface height; u is the height approximation and A_e is the element area. This error measure $\bar{E}_e$ is the modified element error (5): E_e is divided by the element area and the square root is taken. The error indicator $\bar{E}_e$ is measured in length units and can be treated as some averaged absolute error over an element.

Results of surface approximation by quadtrees of quadratic quadrilateral elements are presented in Figures 5 and 6. Fig. 5 shows element quadtrees for error tolerance values 0.0005 (6 iterations) and 0.0001 (7 iterations). Visualization of the approximated surface (error tolerance 0.0005) with the use of Java 3D is presented in Fig. 6.

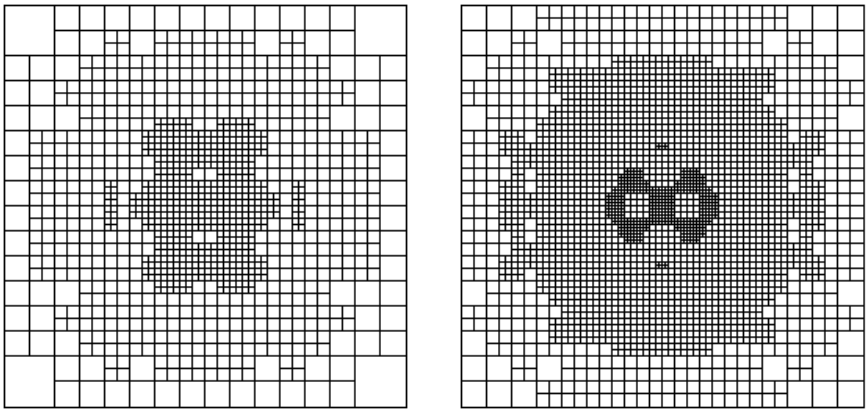


Fig. 5. Quadtrees of quadratic quadrilaterals characterized by different height approximation errors

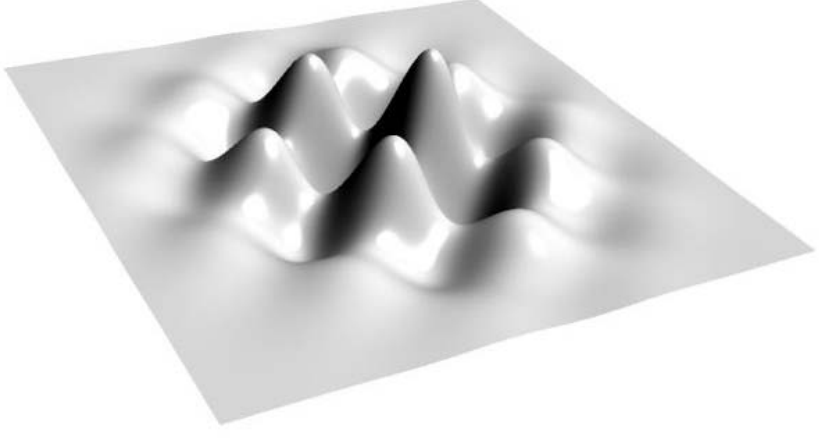


Fig. 6. Visualization of a surface approximated by quadratic quadrilaterals

6 Conclusion and Future Work

We have introduced special quadratic quadrilateral elements for adaptive surface modeling. Two special quadrilateral elements can be connected to one ordinary edge of a quadrilateral quadratic element. The special refinement elements are created by placement of one or two midside nodes outside the element area and by modification of element shape functions. The modified shape functions maintain geometry and field continuity across element T-junctions. It is worth noting that the refinement elements do not differ from standard quadratic finite elements and can be incorporated into existing finite element codes.

Ordinary and special quadratic quadrilateral elements are used for surface approximation. Global approximation error is minimized by solution of a variational problem using the finite element method. A local element error indicator is utilized for adaptive mesh refinement. Elements with excessive local errors are subdivided into four elements each. At any refinement stage the element mesh is topologically equivalent to a quadtree. The quadtree data structure is used to store element data and to navigate through the mesh. Quadtree balancing is performed after each mesh refinement step in order to provide conforming connections of special and ordinary elements.

The proposed algorithm of surface modeling with a quadtree of quadratic quadrilaterals is demonstrated on the surface height approximation for a square domain. A surface mesh produced by the algorithm can be directly used in finite element analysis, where quadrilateral elements are considered more suitable than triangles.

Quadrilateral refinement elements can be employed for general surface modeling and for problems of the "surface on surface" type. For general surface modeling, a coarse starting mesh of quadrilateral elements topologically consistent with the modeled surface is created. Quadtree subdivisions are performed inside each element of the starting mesh. Mesh balancing should include balancing of quadtrees inside each starting element and balancing elements on the interelement boundaries of the starting mesh. Future research will follow this direction.

References

1. Duchineau, M. et al.: ROAMing terrain: real-time optimally adapting meshes. Procs of the IEEE Visualization 97 (1997) 81-88.
2. Pajarola, R.: Large scale terrain visualization using the restricted quadtree triangulation. Procs of the IEEE Visualization 98 (1998) 19-24.
3. Grosso, R., Lurig, C., Ertl, T.: The multilevel finite element method for adaptive mesh optimization and vizualization of volume data. Procs of the IEEE Visualization 97 (1997) 387-395.
4. Bathe, K.-J.: Finite Element Procedures. Prentice-Hall (1996).
5. Frey, P.J., George, P.-L.: Mesh Generation. Application to Finite Elements. Hermes (2000).
6. Zorin, D., Schröder, P.: Subdivision for modeling and animation. SIGGRAPH'00 Course Notes (2000).
7. Kobbelt, L.P.: A subdivision scheme for smooth interpolation of quad-mesh data. Procs of EUROGRAPHICS'98, Tutorial (1998).
8. Fortin, M., Tanguy, P.: A non-standard mesh refinement procedure through node labelling. Int. J. Numer. Meth. Eng. **20** (1984) 1361-1365.
9. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCs. ACM Trans. on Graphics, **22** (2003) 477-484.
10. de Berg, M., van Kreveld, M., Overmars, M. Schwarzkopf, O.: Computational Geometry. Algorithms and Applications. Springer (2000).

MC Slicing for Volume Rendering Applications

A. Benassarou[1], E. Bittar[1], N. W. John[2], and L. Lucas[1,⋆]

[1] CReSTIC / LERI / MADS
Université de Reims Champagne-Ardenne, Reims, France
[2] School of Informatics,
University of Wales, Bangor, United Kingdom

Abstract. Recent developments in volume visualization using standard graphics hardware provide an effective and interactive way to understand and interpret the data. Mainly based on 3D texture mapping, these hardware-accelerated visualization systems often use a cell-projection method based on a tetrahedral decomposition of volumes usually sampled as a regular lattice. On the contrary, the method we address in this paper considers the slicing problem as a restricted solution of the marching cubes algorithm [1, 2]. Our solution is thus simple, elegant and fast. The nature of the intersection polygons provides us with the opportunity to retain only 4 of the 15 canonical configurations defined by Lorensen and Cline and to propose a special look-up table.

1 Introduction

Interactivity is often regarded as a necessary condition to efficiently analyze volumetric data, and so obtaining fast enough rendering speeds has historically been a major problem in volume visualization systems. Over the last decade, a large number of methods have been developed to significantly improve traditional approaches, which are known to be very expensive with respect to CPU usage [3]. The generalized use of modern PC graphics boards is part of these recent advances to offer today's users a good level of interactivity [4, 5, 6].

The volume rendering algorithm that we have developed employs a novel technique that is centered on an efficient incremental slicing method derived from the marching cubes algorithm [1, 2]. This approach allows us to achieve interactive rendering on structured grids on standard rendering hardware. In this paper we present a brief overview of related work (Sec. 2), and review the basic concepts of direct volume rendering via 3D texture mapping (Sec. 3). We then describe in detail our incremental slicing algorithm together with an analysis of the results that we have obtained from our visualization system.

⋆ Correspondence to: laurent.lucas@univ-reims.fr, Rue des Crayères, BP 1035, 51687 Reims Cedex 2.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 314–321, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Background and Related Work

The two main categories of volume visualization techniques in popular use are surface extraction algorithms and direct volume rendering algorithms. The key idea of surface-based rendering is to extract an intermediate surface description of the relevant objects from the volume data [2]. In volume rendering, images are created directly from the volume data, and no intermediate geometry is extracted [7]. Our work is concerned with this second category, and in particular, with *interactive* volume rendering methods. Note, however, that we will make use of a technique first developed for the marching cubes surface-based rendering algorithm.

A variety of software and hardware approaches are possible to implement direct volume rendering. They will typically employ one of two basic scanning strategies for traversing the volume:

Feed Backward Projection or *Image Order Traversal.* The pixels in the image plane are traversed and imaginary rays are cast through each pixel in the volume. The path of the ray determines the pixel value. Lavoy's raycasting algorithm is an example of image order traversal.
Feed Forward Projection or *Object Order Traversal.* The data volume is traversed and each voxel in the volume is projected onto the image plane. Splatting [8] is a good example of an Object Order Traversal technique.

These strategies correspond to the image and object order rasterization algorithms. High quality volume rendering, at the cost of compute time, is provided by raycasting and splatting; lower quality, but faster rendering, is provided by shear-warp rendering [9] and texture-based methods. These faster methods will typically make use of graphics hardware to provide interactive rendering rates, and the latest generations of commodity graphics cards, such as NVidia GeForce and ATI Radeon families, are proving to be ideal for this purpose. In the following sections, we have classified algorithms that make use of commodity graphics cards into: projection-based methods and slicing-based methods. We then provide an overview of texture mapping methods for volume rendering.

Projection-based methods. Shirley and Tuchman [10] were amongst the first to use polygon rendering hardware support for approximate volume rendering. Based on a decomposition into tetrahedra of any part of three-dimensional data, the projected tetrahedra (PT) algorithm proceeds first by classifying each tetrahedron according to its projected profile in order to find the positions of the tetrahedra vertices after the perspective transformation and to decompose them into triangles. This idea of the PT algorithm has subsequently been re-used by many similar works since then. For example, Stein *et al.* [11] attempted to improve these approximations by employing a more accurate sorting algorithm.

Slicing-based methods. Slicing-based methods can be considered as an approximation of the previous methods, whereby the projections of the faces of a polyhedral element are approximated by a set of polygons. Yagel *et al.* [12] proposed

a technique that allows the faces to be approximated by a polygon that represents its intersection with a sweep plane. They show that this technique can render visually comparable images faster without having to explicitly store any kind of vertex or face adjacency information which is necessary for most other methods. Other proxy geometry, such as spherical shells, may be used to eliminate artifacts caused by perspective projection [13]. More recently, Chopra and Meyer [14] have improved Yagel's incremental slicing method whereas Lensch *et al.* [15] have proposed a new paradigm based upon a slicing prism.

3 Slicing-Based Methods for Hardware Texture Mapping

The OpenGL application programming interface provides access to the advanced per-pixel operations that can be applied at the rasterization stage of the graphics pipeline, and in the frame buffer hardware of modern graphics workstations. In particular, they provide sufficient power to render high resolution volume data sets with interactive frame rate using 2D or 3D texture mapping.

Object-aligned slicing. Support for 2D texture mapping is now a standard feature of modern graphics PCs, and is suitable for implementing object-aligned slicing. The principle is similar to the shear-warp algorithm [9]. It involves storing a set of three rectilinear volume data sets, and using them as three perpendicular stacks of object aligned texture slices (Fig. 1). Slices are taken through the volume orthogonal to each of the principal axes and the resulting information for each slice is represented as a 2D texture that is then pasted onto a square polygon of the same size. The rendering is performed by projecting the textured quads and blending them back-to-front into the frame buffer. During the process of texture mapping the volume data is bilinearly interpolated onto a slice polygon.

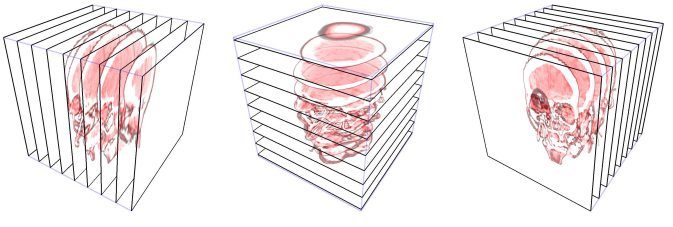


Fig. 1. Object-aligned slice stacks with 2D texture mapping

View-aligned slicing. The use of 3D texture mapping hardware has become a powerful visualization option for interactive high-quality direct volume rendering [16, 6]. The rectilinear volume data is first converted to a 3D texture. Then, a number of planes perpendicular to the viewer's line of sight are clipped against the volume bounding box. The texture coordinates in parametric object space

are assigned to each vertex of the clipped polygons. During rasterization, fragments in the slice are trilinearly interpolated from 3D texture and projected onto the image plane using adequate blending operations (Fig. 2).

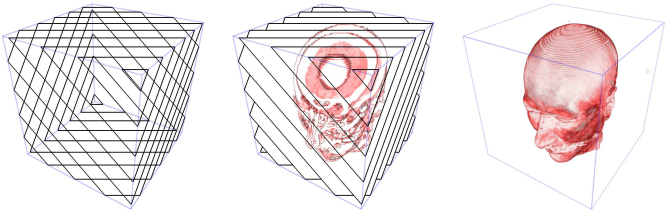


Fig. 2. View-aligned slice stacks with 3D texture mapping

Proxy geometry characterization. The proxy geometry characterization step in the graphical pipeline can be specified by either enclosing rectangles of intersections, or polygons of intersections. The use of enclosing rectangles is a straightforward method of texture mapping cut-planes. Other approaches are more complex and require finding the polygon of intersection between a given cut-plane and the cube of data. Directly performed on the CPU or the GPU, this approach is faster for processing fragments, because one visits only those fragments that are inside the cube of data. The approach used by Kniss *et al.* [17] considers the following stages:

1. Transform the volume bounding box vertices into view coordinates.
2. Find the minimum and maximum z coordinates of the transformed vertices.
3. For each plane, in back-to-front order:
   (a) Test for intersections with the edges of the bounding box and add each intersection point (up to six) to a fixed-size temporary vertex list.
   (b) Compute the centre of the proxy polygon by averaging the intersection points and sort the polygon vertices clockwise [18].
   (c) Tessellate the proxy polygon into triangles and add the resulting vertices to the output vertex array.

Unfortunately, this algorithm suffers from its re-ordering stage. Conversely, the method we propose provides an implicitly ordered sequence of vertices that can be directly drawn by OpenGL function calls. This novel, easy to use algorithm is described below.

4 MC Slicing Algorithm

Marching cubes principle. Concisely, the original marching cubes algorithm allows one to efficiently polygonize an approximation of the intersection between a surface and a cube. The approximation is achieved through evaluating some predicate at the eight corners of the cube. The 256 possible solutions are known and stored in a precalculated table. Each entry of this table is a triplet sequence which indicates the edges hit by the surface and allows us to interpolate the intersection triangles.

Adjustments for slicing purposes. In our case, we have a singular surface and a single cube. The surface is a view-aligned plane, and the cube is the 3D texture. The predicate we test on each corner of this unit cube is defined as follows: "Is this corner deeper than the current plane?" (Algo. 1, line 1). When the intersection between the cube and the plane is not empty (0 vertex), it is either a triangle (3 vertices), a quad (4), a pentagon (5) or a hexagon (6). As there is never more than one connected component, the 256 surfaces might be stored directly as polygons (Table 1) instead of triangle sets. In fact, these polygons are all convex and they can even be transmitted to OpenGL as triangle fans.

Algorithm 1. McSliceCube ()

```
   begin
       for i ∈ [0...8[ do
           Z_i = (i&1 ≠ 0 ? GL_MODELVIEW_{0,2} : 0)
               + (i&2 ≠ 0 ? GL_MODELVIEW_{1,2} : 0)
               + (i&4 ≠ 0 ? GL_MODELVIEW_{2,2} : 0) + GL_MODELVIEW_{3,2};
       z_min = min_{i∈[0...8]}(Z_i);
       z_max = max_{i∈[0...8]}(Z_i);
       for z ∈ [z_min ... z_max] do
1          key = Σ_{i∈[0...8[} (Z_i > z ? 2^i : 0);
           if T_{key,0} ≠ −1 then
               glBegin (GL_TRIANGLE_FAN);
2              McSliceEdge (Z, z, T_{key,0});
3              McSliceEdge (Z, z, T_{key,1});
4              McSliceEdge (Z, z, T_{key,2});
5              for (i = 3; T_{key,i} ≠ −1; i = i + 1) do McSliceEdge (Z, z, T_{key,i});
               glEnd ();
   end
```

Algorithm 2. McSliceEdge (Z, z, e)

```
   // Vertices + coordinates of an edge
   edge_v0, edge_v1 : EdgeId ↦ VertexId
   edge_x, edge_y, edge_z : EdgeId ↦ ∅ ∪ {0, 1}
   begin
       z_0 = Z[v_0] where v_0 = edge_v0 (e);
       z_1 = Z[v_1] where v_1 = edge_v1 (e);
       r = (z − z_0)/(z_1 − z_0);
       switch e do
           case 0, 1, 2, 3 : glVertex3d (r, edge_y (e), edge_z (e));
           case 4, 5, 6, 7 : glVertex3d (edge_x (e), r, edge_z (e));
           case 8, 9, 10, 11 : glVertex3d (edge_x (e), edge_y (e), r);
   end
```

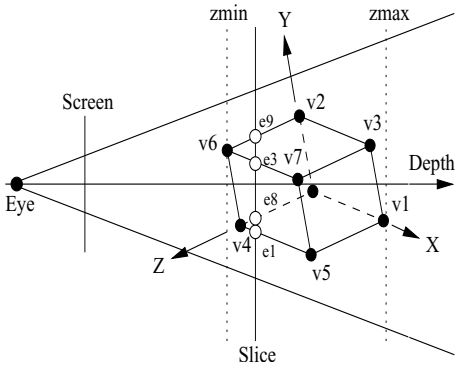


Fig. 3. Example of MC slicing. Vertices 0, 1, 2, 3, 5 and 7 are deeper than the slicing plane

Table 1. Excerpt from our table. Each entry is an ordered sequence of edges hit by the surface

$\mathbb{T}$	*intersected edge sequence*
0	-1, -1, -1, -1, -1, -1, -1, -1
1	4, 8, 0, -1, -1, -1, -1, -1
⋮	⋮
174	1, 3, 9, 4, 0, -1, -1, -1
175	3, 9, 8, 1, -1, -1, -1, -1
176	3, 11, 10, 8, 5, -1, -1, -1
⋮	⋮
254	4, 0, 8, -1, -1, -1, -1, -1
255	-1, -1, -1, -1, -1, -1, -1, -1

Example. If we consider the case seen in Fig. 3, we observe that six vertices are deeper than the current cut plane. Those vertices are named 0, 1, 2, 3, 5 and 7. Line 1 of Algo. 1 gives $1 + 2 + 4 + 8 + 32 + 128 = 175$ as index of the first dimension of Table 1. At this index, we find that the ordered sequence $\mathbb{T}_{175} = \{3, 9, 8, 1, -1, -1, -1, -1\}$, which means that the intersection is not empty ($\mathbb{T}_{175,0} \neq -1$) and the first edge to be processed is edge number 3 ($\mathbb{T}_{175,0}$). Line 2 calls Algo. 2 which performs the intersection for this edge (interpolation between the two ends of the edge, v_6 and v_7). Similarly, lines 3 and 4 compute the intersection points with edges 9 ($\mathbb{T}_{175,1}$) and 8 ($\mathbb{T}_{175,2}$). Because there is a fourth edge ($\mathbb{T}_{175,3} \neq -1$), we then enter the loop and finally operate line 5 (edge 1). The algorithm ends the triangle fan here since $\mathbb{T}_{175,4} = -1$.

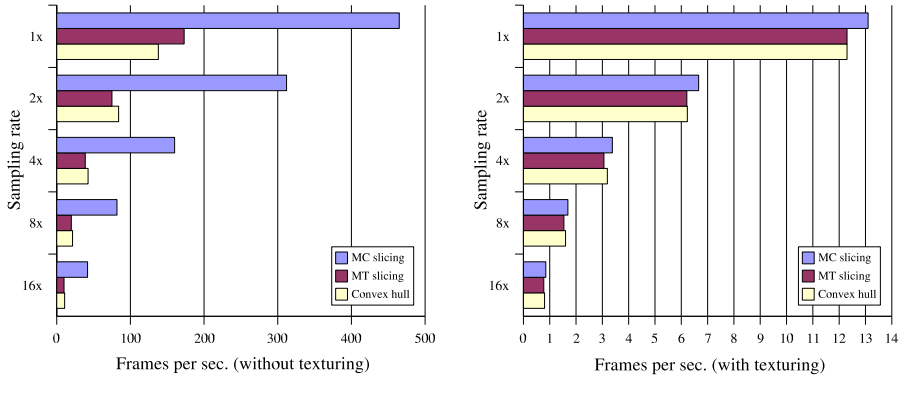


Fig. 4. Comparisons between our methods and the usual Convex hull approach

5 Results and Discussion

For comparison purposes, we have implemented the usual Convex hull solution (Sec. 3). We have also developed a marching tetrahedra-like algorithm. Whereas MC slicing operates directly on a cube, MT slicing decomposes the same cube into six tetrahedra and computes the intersection between the slicing plane and each tetrahedron. There are fewer cases (16 instead of 256) and intersections are either triangles or quads when they exist. The main advantage offered by the original marching tetrahedra is that the resulting geometry does not suffer from the possible ambiguities of the marching cubes. Nevertheless, the simplicial decomposition involves more OpenGL primitives as we compute six intersections instead of one. Because there cannot be any ambiguity when intersecting a plane and a cube, we consider that the extra computational cost is not really worth it. Figure 4 presents the comparison between the three discussed methods. The performance measurements were obtained on a Linux platform equipped with an AMD Athlon XP 2200+ CPU and a GeForce 6800 TD graphic board using a viewport size of 704×576 pixels. The volume data ($512 \times 512 \times 106$ CT scan) is illustrated on Fig. 2. Each technique has been run five times at five different sampling rates : $1\times$, $2\times$, $4\times$, $8\times$ and $16\times$ (distance between slices $= 1/16$). Algorithm 2 has also been coded with shading languages such as Cg or GLSL but we did not notice any real gain. The benchmarks present the number of frames per second reached without and with actual texturing process. We observe that, at low sampling rates, our method shortly accelerates the rendering. The real impact of our method can be observed with higher sampling rates: from $4\times$ to $16\times$, the MC algorithm performs the same slicing as the other two approximately four and five times quicker. This major improvement in the performance is mainly due to the simplicity of the algorithm. Like the original marching cubes, our method owes its efficiency to the precalculation of a 2 Kbytes look-up table.
In summary, the two major advantages of the MC slicing approach are: it processes the whole cube without any tetrahedral decomposition; and it generates the surface vertices directly in the correct order. These advantages allow us to save on the CPU time and to achieve higher frame rates.

6 Conclusion

In this paper, we presented an accelerated slicing algorithm for interactive volume rendering of structured grids. Derived from the classic marching cubes, it requires a small amount of memory and provides adaptive rendering for improved image accuracy as well as progressive rendering for rapid feedback at interaction time. It is finally suited to exploit graphics hardware. There is a growing requirement for interactive volume visualization from medical applications. Collaborative work is beginning on developing a virtual reality simulator for interventional radiology procedures [19], where fast and efficient rendering of patient specific data is a major requirement. We intend using the MC slicing algorithm to provide this requirement. This will enable us to further develop and refine the ideas presented in this paper.

References

1. Wyvill, B., Wyvill, G., McPheeters, C.: Data structure for soft objects. The Visual Computer **2** (1986) 227–234
2. Lorensen, W., Cline, H.: Marching cubes : a high resolution 3D surface construction algorithm. Computer Graphics **21** (1987) 163–169
3. Brodlie, K., Wood, J.: Recent advances in visualization of volumetric data. In: Proc. Eurographics 2000 - STAR Reports. (2000) 65–84
4. Engel, K., Ertl, T.: High-quality volume rendering with flexible consumer graphics hardware. In: Proc. Eurographics '02 - STAR Reports. (2002)
5. Roettger, S., Guthe, S., Weiskopf, D., Ertl, T., Strasser, W.: Smart hardware accelerated volume rendering. In: Proc. Eurographics/IEEE TCVG Symposium on Visualization. (2003) 231–238
6. Westermann, R., Ertl, T.: Efficiently using graphics hardware in volume rendering applications. Computer Graphics **32** (1998) 169–179
7. Levoy, M.: Display of surfaces from volume data. IEEE Computer Graphics and Applications **8** (1988) 29–37
8. Westover, L.: Footprint evaluation for volume rendering. Computer Graphics **24** (1991)
9. Lacroute, P., Levoy, M.: Fast volume rendering using a shear-warp factorization of the viewing transformation. Computer Graphics **28** (1994) 451–458
10. Shirley, P., Tuchman, A.: A polygonal approximation to direct scalar volume rendering. Computer Graphics **24** (1990) 63–70
11. Stein, C., Becker, B., Max, N.: Sorting and hardware assisted rendering for volume visualization. In: Proc. ACM Symposium on Volume Visualization. (1994) 83–90
12. Yagel, R., Reed, D., Law, A., Shih, P., Shareef, N.: Hardware assisted volume rendering of unstructured grids by incremental slicing. In: Proc. ACM Symposium on Volume Visualization '96. (1996) 55–63
13. LaMar, E., Hamann, B., Joy, K.: Multiresolution techniques for interactive texture-based volume visualization. In: Proc. ACM Symposium on Volume Visualization '99. (1999) 355–361
14. Chopra, P., Meyer, J.: Incremental slicing revisited: Accelerated volume rendering of unstructured meshes. In: Proc. IASTED Visualization, Imaging and Image Processing '02. (2002) 533–538
15. Lensch, H., Daubert, K., Seidel, H.: Interactive semi-transparent volumetric textures. In: Proc. Vision, Modeling and Visualization '02. (2002) 505–512
16. Cabral, B., Cam, N., Foran, J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In: Proc. ACM Symposium on Volume Visualization '94. (1994) 91–98
17. Kniss, J., Kindlmann, G., Hansen, C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: Proc. Visualization '01. (2001) 255–262
18. Moret, B., Shapiro, H.: Algorithms from P to NP. Volume I: Design and Efficiency. Benjamin-Cummings (1991)
19. Healey, A., Evans, J., Murphy, M., Gould, D., Phillips, R., Ward, J., John, N., Brodlie, K., Bulpit, A., Chalmers, N., Groves, D., Hatfield, F., How, T., Diaz, B., Farrell, M., Kessel, D., Bello, F.: Challenges realising effective radiological interventional virtual environments: the CRaIVE approach. In: Proc. Medicine meets Virtual Reality, IOS Press (2004) 127–129

Modelling and Sampling Ramified Objects with Substructure-Based Method

Weiwei Yin[1], Marc Jaeger[2], Jun Teng[1], and Bao-Gang Hu[1]

[1] Institute of Automation, Chinese Academy of Sciences, China
[2] CIRAD, AMAP, France
{wwyin, jaeger, jteng, hubg}@liama.ia.ac.cn
http://liama.ia.ac.cn

Abstract. This paper describes a technique that speeds up both the modelling and the sampling processes for a ramified object. By introducing the notion of substructure, we divide the ramified object into a set of ordered substructures, among which only a part of basic substructures is selected for implicit modelling and point sampling. Other substructures or even the whole object can then be directly instantiated and sampled by simple transformation and replication without resorting to the repetitive modelling and sampling processes.

1 Introduction

Smooth blending junctions and complex structures are two distinct characters for many ramified objects. Some parametric approaches [1] [9] have been explored to model and visualize such objects. While, implicit methods [2] [3] which we would prefer more, well exhibit the local details in ramification due to their unmatched advantages in generating smooth blending surface.

However, implicit surface has its own difficulty in generating sampling points and surface reconstruction when rendering. Bloomenthal and Wyvill first proposed to use scattered seed points to sample implicit surface in [2]. After that, more attentions were paid on the physically based sampling algorithms in [4] [5]. Witkin implemented a quite complete particle system to sample blobby spheres and cylinders in [6].

In this paper, we do aim to efficiently model and sample such ramified object with smooth blending junctions. With the precondition that the ramified object can be hierarchically organized, we recursively divide the object into a set of ordered substructures, among which only a part of basic substructures is selected as the new target for implicitly modelling and basic point sampling. Other substructures or even the whole object can then be directly instantiated and sampled by transforming and replicating sampling points of the new target without real geometric modelling and sampling processes.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 322–326, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Object Decomposition

We consider a ramified object consisting of a *main branch* and several *lateral branches*, which may have other *lateral branches* on them. Based on the botanical knowledge, we separate the *lateral branch* from the *main axis* and divide this ramified object into several types of similar subparts (substructures) [7]. Each substructure is assigned a hierarchical order and redivided in the same way until it reaches the basic form Sub_{basic} as:

$$Sub_{basic} = \{Axis + Skeleton_0 + \cdots + Skeleton_i + \cdots + Skeleton_m\} \quad (1)$$

where $Skeleton_i$ denotes a single *lateral branch* attached to the *main axis* $Axis$ and can not be redivided.

An example of two-step decomposition for a ramified object is shown in the following equations:

$$Ojbect = \{Axis_{main} + Sub_1 + \cdots + Sub_i + \cdots + Sub_n\} \quad (2)$$

$$Sub_i = \{Axis_{imain} + Sub_{i1} + \cdots + Sub_{ij} + \cdots + Sub_{ik}\} \quad (3)$$

where $Axis_{main}$ and $Axis_{imain}$, represented by a single branch, denote the corresponding *main branches*, Sub_i denotes the substructure in hierarchy 1, Sub_{ij} denotes the substructures in hierarchy 2 that constitute Sub_i in hierarchy 1.

Since our decomposition is fully based on the connective relationship between the *main axis* and the *lateral branch*, a neighborhood graph for substructures either of the upper-lower hierarchies or among the same hierarchy can be easily built up. Additionally, level of detail (LOD) techniques can be conveniently applied in such defined system because all substructures in each order can be quickly reached and retrieved through the neighborhood graph.

3 Implicit Modelling

A common and intuitive method for modelling smoothly blending surface is based on the underlying skeletons [2] [3]. A skeleton-based implicit surface is defined as a set of points $P(x, y, z)$ as follows:

$$Surface = \{P(x, y, z) \in \Re^3 \mid f(P) = 0\} \quad (4)$$

where $f(x, y, z)$ is a field function defined by the skeleton.

For a branching structure consisting of skeletons $s_i (i = 1, \cdots, n)$ with the associated field functions $f_i (i = 1, \cdots, n)$, we define such a skeleton-based implicit surface and apply blending techniques, for instance the simplest sum operation $f(P) = \sum_{i=1}^{n} f_i(P)$, to generate smooth blending surface.

4 Point Sampling

The basic idea of point sampling is inspired by Witkin [6], who generated a well distributed sample of particles while had high computational cost when the sampling particle number was huge. In order to improve sampling efficiency as well as remain the high sampling density, we propose a *substructure-based* method, which uses a part of the ramified object instead of the whole one as the sampling target. Since the target is shrunk and simplified, the sampling particle number is reduced, followed by the computational complexity.

Six steps are included in our *substructure-based* sampling method and each step is briefly described in the following:

Step 1. Define a new target by selecting a representation from each type of basic substructures.

Step 2. Model the new target with skeleton-based implicit surfaces.

Step 3. Apply Witkin's sampling algorithm [6] to the implicitly modelled surfaces of the new target. Relevant information of sampled particles, including 3D positions and surface normals, is stored.

Step 4. Instantiate other substructures and the whole object by directly involving and linear geometric transforming the sampled points of the new target.

Step 5. Detect ramiform particles which come from different substructures and penetrate each other as shown in Fig.1 (left). A particle $P(x, y, z)$ is considered as a ramiform particle only when
- It is inside the ramification area between substructures, and
- The field function value at it satisfies: $|f(P)| > \varepsilon$

where ε is a very small positive number.

Step 6. Delete all ramiform particles and apply Witkin's sampling algorithm again to the skeletons which lie inside the corresponding ramification. A new blending surface will be quickly generated as shown in Fig.1 (left-center).

Non-ramiform particles are considered static on their own position. Sampling time and computational cost are greatly reduced without managing all particles at each iteration even if the total number of the particles is very huge. Moreover, since only ramifications existing between upper-lower substructures are considered, no unwanted blending will exist between actually non-connected branches.

5 Implementation and Results

The techniques described in this paper have been implemented in C++ code. The groundwork of our present experiments is the *GreenLab Model* [8] whose output is a set of hierarchically structured line skeletons. A simple exponential function is defined as the field function f_i for the ith line skeleton:

$$f_i(P) = \exp[-\frac{d_i(P)^2}{R_i^2} + 1] - 1 \tag{5}$$

where $d_i(P)$ is the algebraic distance of point $P(x, y, z)$ to the ith skeleton. R_i, a radius parameter, may be a constant for a constant-radius branch or be computed by certain linear variable function for a tapered branch.

Some simple examples are shown below. A substructure composed of two kinds of basic forms is shown in Fig.1 (center). Moreover, with the original one shown in Fig.1 (right-center), the result of adding a new branch in Fig.1 (right) is quickly achieved with no need to sample all the skeletons again.

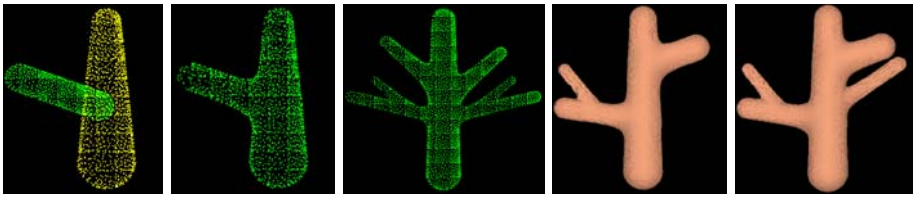


Fig. 1. Left-Right: Before merging (left); After merging (left-center); An example of substructures (center); Before adding a new branch (right-center); After adding a new branch (right)

6 Conclusions and Future Work

In this work, we have proposed a substructure-based method to implicitly model and sample ramified objects. With the shrunk and simplified target and the utilization of substructures, both the modelling and sampling processes are sped up and repetitious works for modelling and sampling the same type of substructures are avoided. As for the future works, the proposed method will be applied to real ramified objects. A texture algorithm for sampling particles is also required.

Acknowledgement

This work is supported in part by the national Natural Science Foundation of China (#60073007, #60473110) and by China 863 Program (#2002AA241221).

References

1. Bloomenthal, J.: Modeling the Mighty Maple. In: Proc.SIGGRAPH'85, Vol. 19. (1985) 305–311
2. Bloomenthal, J., Wyvill, B.: Interative Techniques for Implicit Modeling. In: Computer Graphics Forum, Vol. 24. (1990) 109–116
3. Bloomenthal, J., Shoemake, K.: Convolution Surfaces. In: Proc.SIGGRAPH'91, Vol. 25. (1991) 251–256
4. Turk, G.: Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion. In: ACM TOG, Vol. 25. (1991) 289–298
5. de Figueiredo, L.H., Gomes, J.: Sampling Implicit Objects with Physically-Based Particle Systems. In: Computer Graphics, Vol. 20. (1996) 365–376

6. Witkin, A.P., Heckbert, P.S.: Using Particles to Sample and Control Implicit Surfaces. In: Proc.SIGGRAPH'94. (1994) 269-277
7. Yan, H.P., Barczi, J.F., de Reffye, P., Hu, B.-G.: Fast Algorithms of Plant Computation Based on Substructure Instances. In: Proc.the 10th International Conference in Central Europe on Computer Graphics'02, Vol, 10. (2002) 145–153
8. Yan, H.P., Kang, M.Z., de Reffye, P., Dingkuhn, M.: A Dynamic Architectural Plant Model Simulating Resource-Dependent Growth. In: Annals of Botany, Vol, 1. (2004) 591–602
9. Felkel, P., Kanitsar, A., Fuhrmann, A., Wegenkittl, R.: Surface Models of Tube Trees. In: Computer Graphics International'04. (2004) 70–77

Integration of Multiple Segmentation Based Environment Models

SeungTaek Ryoo[1] and CheungWoon Jho[2]

[1] Department of Software, HanShin University, Korea
stryoo@hs.ac.kr
[2] Division of Digital Contents, DongSeo University, Korea
cwjho@dreamwiz.com

Abstract. An environment model that is constructed using a single image has the problem of a blurring effect caused by the fixed resolution, and the stretching effect of the 3D model caused when information that does not exist on the image occurs due to the occlusion. This paper introduces integration method using multiple images to resolve the above problem. This method can express parallex effect and expand the environment model to represent wide range of environment. [3]

1 Introduction

The image-based modeling method is the one that is being studied the most due to its efficiency. This method enables real-time rendering because it extracts the geometric information from the image that represent the environment in the pre-processing stage. The objective of this paper is on real-time rendering for free navigation as realistically as possible. A method for acquiring the depth-image through image segmentation is suggested to construct the environment model. Also, the environment model has been made to be expandable through registration and integration of the multiple environment models.

The methods using the planes[1, 2, 3] reconstruct the 3D model based the reference plane. Methods using the vanishing point and vanishing line[1, 2] and the image editing method through interaction[3] are some examples of this method. The first method performs the modeling of the environment using the vanishing point based on the acquired plane, which makes it difficult to acquire an accurate environment model. The second method enables a more accurate modeling but requires user interaction using various tools. Horry[1] used a plane constructed using spider mesh to set the depth value and Criminisi[2] took into account the relationship between the plane parallel to the reference plane and the vanishing point in calculating the distance. Oh[3] used the floor parallel tool and the perpendicular tool to set the location on the reference plane. To resolve the

[3] This work was supported by the Korea Research Foundation Grant(KRF-2004-003-D00339).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 327–330, 2005.
© Springer-Verlag Berlin Heidelberg 2005

problem mentioned above, we used an environment modeling method based on depth image acquired through image segmentation[4]. This method makes the environment modeling easier and can be implemented on an environment map.

2 Multiple Environment Models

A 3D model constructed using a single image has a problem of stretching effect when occluded object appear on the scene. A 3D environment model based on multiple images is required to resolve this problem. To do this, the integration method based on the corresponding line is suggested in this paper. the process of integrating the multiple environment models is as follows. First of all, the images acquired are reconstructed into 3D environment models using the method shown in previous chapter. Then the corresponding points are set from the image to be integrated and the 3D environment models are registered by using the transformation (translation, rotation, scaling) through the corresponding points. To acquire a more accurate environment model, the subdivision method is applied to integrate the environment models. Finally, the texel values are mixed and recreated to resolve the texture inconsistency effect to acquire a desired image.

2.1 The Registration of the Environment Models

Each image is divided into a floor, ceiling and surrounding objects[4]. The 3D environment models acquired through different viewpoints have the characteristic that the reference plane equation is equal to each other. By using this characteristic, the 3D coordinate registration can be simplified into a 2D coordinate registration. This means that the reconstructed environment models share the planes that form the floor and the ceiling which makes the registration of each environment model an easy task through 2D transformation. Figure 1 shows the process of integrating the 3D environment models using two corresponding point. Figure 1-a shows the setting of the two corresponding points from the image by user. The dotted line (corresponding line) indicates the vectors created by the first and second corresponding points. Figure 1-b shows the result image of the environment model E_a created by the left image of Figure 1-a that has been translated by the environment model E_b created by the right image. Figure 1-c shows the environment models created by rotating environment model E_a around the first corresponding point. Figure 1-d shows the two environment models that have been created by scaling environment model E_a. We can see that the environment models can be registered by translating, rotating and scaling the models using only the two corresponding points.

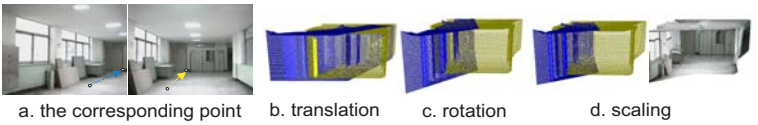


Fig. 1. the registration of the environment model

2.2 The Integration of the Environment Models

we proposed the method of integrating the environment models after partition using a subdivision plane based on a corresponding line. Since two environment models based on only the two corresponding points have been registered, the environment models that are far away from the corresponding line have a larger displacement than those nearby. Therefore, it becomes difficult to discard the redundant areas through polygon comparison and analysis. We have integrated the two environment models using this feature between the corresponding line and the environment model. The environment model has been subdivided into two sub-models using the plane created based on the corresponding line and have chosen the sub-model with less redundant area to form an integrated environment model. Figure 2 shows the integration process of the environment model using the subdivision plane. Figure 2-a shows the process of selecting the area partitioned using the normal of the subdivision plane. Each dotted line indicates the intersection point between each subdivision plane and the environment model and the arrows indicate the direction of the normal of the plane. The sub-models are selected through this direction. Figure 2-b shows the environment models cut by each subdivision plane and Figure 2-c shows the environment model created by integrating the sub-models that have been partitioned. As shown in this figure, this method can be easily integrated the environment models to select and assemble the subdivided models. However, the model created during the integration of the two environment models has a problem of inconsistency in the connection area. To remove this seam, the texture samples of the two environment models that overlap each other must be blended for use. We have used the corresponding line for the consistency of the images in this paper. The seam can be fully removed by repeatedly applying texel blending.

Fig. 2. the integration of the environment models using the subdivision plane

3 The Results

Figure 3 shows the process of constructing an environment model from three regular images acquired from different viewpoints and angle. In *model*1, the view orientation is towards the center of the hall, *model*2 to the left-side, and *model*3 to the right side to form the environment model. The two models, *model*2 and 3, are merged around *model*1. The merged models are divided into two sub-models using the subdivision plane, and the sub-model with less redundancy is selected for the integration. A seam is found on the connection area of an integrated

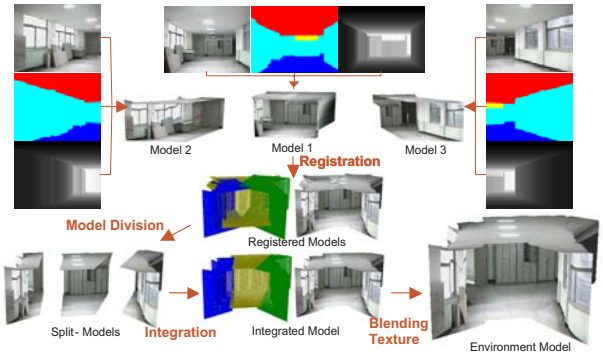


Fig. 3. the construction of the environment model using the multiple images

environment model. The environment model is reconstructed by blending the related texels from each texture map.

4 Conclusion and Future Work

The environment modeling method through the multiple images is able to create precise environment model that can freely change the viewpoint and has an optimal resolution image. It can also extend the navigation area through the integration of the environment model by adding a new environment image. The suggested environment modeling method using the multiple images has advantages as well as limitations which require further research. the 3D environment model acquired using the segmentation-based environment model is hard to merge and integrate precisely. Further studies will focus on a new way of acquiring a more precise 3D environment model without using the range image and also on ways for merging and integrating these new environment models automatically.

References

1. Y. Horry, K. Anjyo, K. Arai, "Tour Into the Picture: Using a Spidery Mesh Inter-face to Make Animation from a Single Image", SIGGRAPH '97, pp 225-232, 1997
2. A. Criminisi, I. Reid, A. Zisserman, "Single View Metrology", Int. J. of Computer Vision, v.40, n.2, pp. 123-148, 2000
3. Byong Mok Oh, Max Chen, Julie Dorsey, Fredo Durand, "Image-based modeling and photo editing", SIGGRAPH 2001: 433-442
4. SeungTaek Ryoo, "Segmentation Based Environment Modeling Using Single Image", ICIAR2004(LNCS3211), pp 98-105, 2004.

On the Impulse Method for Cloth Animation

Juntao Ye, Robert E. Webber, and Irene Gargantini

The University of Western Ontario, London, Ontario, N6A 5B7 Canada
{juntao, webber, irene}@csd.uwo.ca

Abstract. Computer animation of cloth is often plagued by springs being overstretched. Our approach addresses this problem and presents some preliminary results.

1 Introduction

Cloth animation faces two major challenges: realism and speed. The way these two issues are addressed and resolved largely depends on the model adopted. So far the most successful system for creating realistic folds and creases as cloth is subject to various forces (gravity, for instance) has been the mass-spring model. Here the fabric is represented as a 2D array of nodes, each with a given mass, and each being related to its neighbors by mutual stretching, shearing and bending. Overstretching, however, appears in all the work presented in the literature. It is counteracted, but not completely eliminated, in different ways, such as adjusting nodes position [Pro95], adjusting nodes velocity [VSC01], using momentum transfer [VCM95], and applying impulses to the nodes [BFA02]. We present a new method based on the linearization of a nonlinear formulation of impulse calculation. Applying this new impulse approach to cloth animation solves satisfactorily both overstretching and overcompression. Although not shown here, it turns out that it can be proven [Ye05] that the matrix of the linearized system is symmetric, positive definite. This allows more efficient solvers to be used — thus decreasing the computational burden always present in cloth animation, especially in the instances of collision detection and resolution.

2 The Mass-Spring Model

Our cloth model consists of three kinds of springs: stretching springs to model the response of the cloth when pulled in the direction of its threads; shearing springs to simulate the response of the cloth when pulled diagonally; bending springs to model cloth resistance to out-of-plane forces (see Figure 1 and 2). The stretching spring is linear, while both shearing and bending are angular. The bending spring (not shown) is mounted on the common edge of every pair of adjacent triangles, its movement taking place in a plane normal to both triangles. Note that our bending model is borrowed from Bridson et al. [BMF03], and that the shear force $\mathbf{f}^h$ exerted on three nodes in Figure 1 is defined as

$$\mathbf{f}_i^h = k^h \cos\theta \, \mathbf{u}_i, \text{ for } i = 1, 2, 3$$

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 331–334, 2005.
© Springer-Verlag Berlin Heidelberg 2005

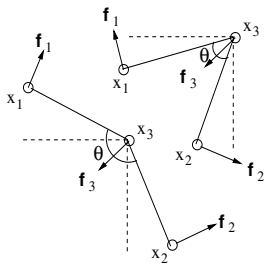


Fig. 1. The shear force model with rest angle $\theta_0 = \pi/2$

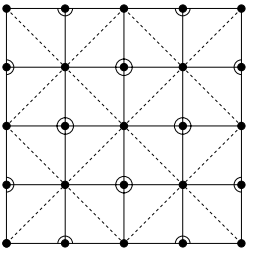


Fig. 2. An isotropic cloth mesh, each right-angle arc represents a shear spring

where k^h is the shear coefficient and

$$\mathbf{u}_1 = \frac{1}{|\mathbf{r}_1|}\frac{\mathbf{r}_1 \times (\mathbf{r}_1 \times \mathbf{r}_2)}{|\mathbf{r}_1 \times (\mathbf{r}_1 \times \mathbf{r}_2)|}, \quad \mathbf{u}_2 = \frac{1}{|\mathbf{r}_2|}\frac{(\mathbf{r}_1 \times \mathbf{r}_2) \times \mathbf{r}_2}{|(\mathbf{r}_1 \times \mathbf{r}_2) \times \mathbf{r}_2|},$$

$$\mathbf{u}_3 = -\mathbf{u}_1 - \mathbf{u}_2, \quad \mathbf{r}_1 = \mathbf{x}_1 - \mathbf{x}_3, \quad \mathbf{r}_2 = \mathbf{x}_2 - \mathbf{x}_3 \ .$$

Thus one angular spring generates three forces (one for each node in the angle). If k^d is the damping coefficient, the shear damping $\mathbf{f}^d$ for these nodes is

$$\mathbf{f}_i^d = -k^d \frac{d\theta}{dt}\mathbf{u}_i, \quad \text{where } \frac{d\theta}{dt} = \mathbf{u}_1 \cdot \mathbf{v}_1 + \mathbf{u}_2 \cdot \mathbf{v}_2 + \mathbf{u}_3 \cdot \mathbf{v}_3 \ .$$

Thus the shearing model is a 2D version of the bending model [BMF03].

3 Constructing the Impulse Equation System

Since $\mathbf{F}\delta t = m\delta\mathbf{v}$, using *impulses* instead of *forces* offers an advantage. Whenever forces are used, in fact, we need to know their magnitude *and* the time during which they are in effect. Impulse, on the other hand, is *directly* related to velocity change. The strain limits for the stretching and compression, L^c and L^s, are set ahead of time by the user. A spring is neither allowed to stretch more than L^s, nor allowed to shrink less than L^c. Whenever $L \in [L^c, L^s]$, the spring tension is a linear function of the spring length according to $\mathbf{f} = k(|\mathbf{x}_j - \mathbf{x}_i| - l_{ij}^0)\frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}$, where l_{ij}^0 is the rest length of the spring connecting x_i and x_j. When a spring is, or potentially will be, out of the limits, impulses are generated and applied to the two end-nodes so that the spring length after the next time step is within the given range. This method works as if there were a massless string and a massless rod passing through the spring (see Figure 3). This string with length L^s is non-stretchable and this rod with length L^c is non-compressible.

A spring x_{ij}, connecting node x_i and x_j with $i < j$, generates impulses I_{ij} to x_i and $-I_{ij}$ to x_j, their directions being collinear with that of $\mathbf{x}_{ij}$. This way, the impulses are considered to be created exclusively by internal forces. When there is no external force acting on a system, both the linear momentum and the

angular momentum of the system are conserved. We define $I_{ij} = s_{ij}\hat{\mathbf{x}}_{ij}$, where $\hat{\mathbf{x}}_{ij} = \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|}$ and s_{ij} is the magnitude of the impulse which we want to compute. Thus node $\mathbf{x}_i$ receives velocity change $\frac{s_{ij}\hat{\mathbf{x}}_{ij}}{m_i}$ and node $\mathbf{x}_j$ receives velocity change $-\frac{s_{ij}\hat{\mathbf{x}}_{ij}}{m_j}$, the positive sign being attributed to the node having smaller index. A node receives impulses from all its incident springs. Consider now seven springs connecting eight nodes, with the node indices satisfying $f < g < h < i < j < k < l < n$ (see Figure 4). The velocity changes for nodes x_i and x_j are

$$\delta\mathbf{v}_i = (s_{ij}\hat{\mathbf{x}}_{ij} + s_{il}\hat{\mathbf{x}}_{il} - s_{fi}\hat{\mathbf{x}}_{fi} - s_{hi}\hat{\mathbf{x}}_{hi})/m_i, \tag{1}$$

$$\delta\mathbf{v}_j = (-s_{ij}\hat{\mathbf{x}}_{ij} - s_{gj}\hat{\mathbf{x}}_{gj} + s_{jk}\hat{\mathbf{x}}_{jk} + s_{jn}\hat{\mathbf{x}}_{jn})/m_j \ . \tag{2}$$

Suppose at time t_0, the spring length is $L^{t_0} = |\mathbf{x}_j - \mathbf{x}_i|$. Once the ODE solver computes the new velocities $\mathbf{v}_i$ and $\mathbf{v}_j$, the nodes will move to new positions accordingly. We can predict the spring length at time $t_0 + h$ to be $\tilde{L}^{t_0+h} = |(\mathbf{x}_j + \mathbf{v}_j h) - (\mathbf{x}_i + \mathbf{v}_i h)|$. If $\tilde{L}^{t_0+h} \not\in [L^{\mathrm{c}}, L^{\mathrm{s}}]$, the spring will be overstretched or overcompressed and we use impulses to change the node velocities so that the new spring length

$$L^{t_0+h} = |\mathbf{x}_j - \mathbf{x}_i + (\mathbf{v}_j + \delta\mathbf{v}_j)h - (\mathbf{v}_i + \delta\mathbf{v}_i)h| \tag{3}$$

satisfies $L^{t_0+h} \in [L^{\mathrm{c}}, L^{\mathrm{s}}]$. We can choose the value for L^{t_0+h} according to the value of $\tilde{L}^{t_0+h}$:

$$L^{t_0+h} = \begin{cases} L^{\mathrm{s}} & \text{if } \tilde{L}^{t_0+h} > L^{\mathrm{s}}; \\ L^{\mathrm{c}} & \text{if } \tilde{L}^{t_0+h} < L^{\mathrm{c}}; \\ \tilde{L}^{t_0+h} & \text{otherwise} \quad . \end{cases}$$

Since each stretching spring corresponds to one equation like Equ. 3, we get a system of nonlinear equations. Using Equ. 1 and 2, Equ. 3 becomes a function of s_{ij} terms. If an appropriate method can solve this nonlinear system of equations, then it is guaranteed that none of the springs will be over-stretched or over-compressed. Methods for solving such nonlinear systems tend to be very slow so a linearization is in order. This approximation is not guaranteed to result in every spring being within the limits after the impulse application, but in our experiments, it always produced springs that are within the limits. Details of the linearization can be found in [Ye05].

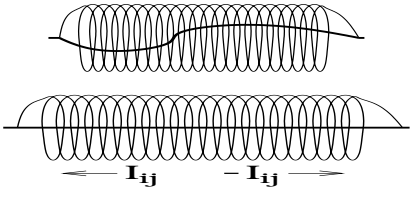


Fig. 3. Spring with string and rod

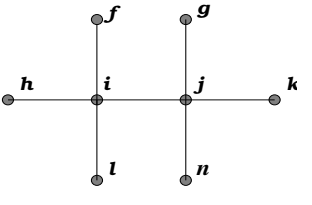


Fig. 4. Seven neighboring springs

Fig. 5. Cloth suspended from two corners

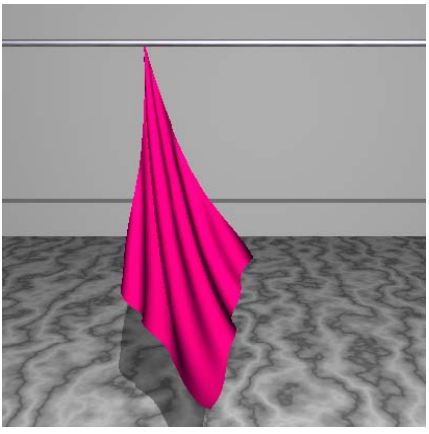


Fig. 6. Cloth swinging down

4 Results and Conclusion

Figure 5 shows the cloth suspended from two corners held 75.2 cm apart. Excessive stretching in mass-spring cloth models would typically appear near the top corners of the cloth. Notice that the cloth around these corners in this figure does not appear overstretched. Figure 6 shows the cloth 1.6 seconds after releasing the top-right corner, while it is still swinging. With the cloth suspended from only one corner, even more force is being applied to the springs at the top-left corner, but they are still not overstretched.

References

[BFA02] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics (SIGGRAPH '02)*, 21(3):594–603, 2002.

[BMF03] Robert Bridson, S. Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2003)*, pages 28–36, 2003.

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pages 147–154, 1995.

[VCM95] Pascal Volino, Martin Courchesne, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of SIGGRAPH '95*, pages 137–144, 1995.

[VSC01] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. In *EUROGRAPHICS '01*, volume 20, pages 260–267, 2001.

[Ye05] Juntao Ye. *Computational Aspects of the Dynamics of Cloth.* PhD thesis, Department of Computer Science, The University of Western Ontario, in progress 2005.

Remeshing Triangle Meshes with Boundaries

Yong Wu, Yuanjun He, and Hongming Cai

Department of Computer Science & Technology, Shanghai Jiao Tong University, China
wuyong916@sjtu.edu.cn

Abstract. This paper proposes a spherical parameterization based remeshing approach to converting a given unstructured triangle mesh with boundaries into one having subdivision connectivity. In order to preserve the boundaries of original meshes, some special strategies are introduced into the remeshing procedure.

1 Introduction

Triangle meshes with subdivision connectivity are important for many multiresolution applications in graphics field. However, most meshes haven't this feature. So there are demands to transform an arbitrary mesh into one with subdivision connectivity. This transformation is called remeshing, which can be understood as an approximation operator $M - S$ that maps from a given irregular mesh M to a regular mesh S with subdivision connectivity. The resulting mesh is called a remesh of the original one. In this section we will give an overview of the most important work.

In [1], Eck et al. have presented a remeshing algorithm. The resulting parameterization is optimal for each base triangle but not smooth across the boundary between two base triangles. Moreover, runtimes for this algorithm can be long due to a lot of harmonic map computations. Lee and co-workers [2] develop a different approach to remeshing irregular meshes. Their method can be used to remesh meshes with boundaries, but the authors don't discuss how to preserve the boundaries.

In this paper, we present an approach for remeshing triangle meshes with boundaries. In order to preserve the boundaries, some special strategies are introduced into the subdividing procedure.

2 Remeshing

2.1 Framework of Our Remeshing Method

As described in Fig.1, our remeshing method contains seven steps.

Step 1: **Closing boundaries.** Before mapping M onto the unit sphere, we triangulate all boundary regions to generate a genus-0 triangle mesh M_Ψ.

Step 2: **Spherical parameterization**. After obtaining M_Ψ, we use Praun's method [3] on M_Ψ to generate a spherical parameterization mesh.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 335 – 338, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Step 3: **Simplifying.** We iteratively execute half-edge contraction operation on C_Ψ to generate the initial base mesh C_t^0.

Step 4: **Optimizing.** To reduce the distortion of the remesh, we insert some new interior vertices to C_t^0 so as to obtain an optimal base mesh C^0 with triangles of similar size.

Step 5: **Subdividing.** The subdivision operation is iterated on C^0 until the error between S_Ψ^m and M_Ψ is below the user specified threshold ε. Here S_Ψ^m is the remesh corresponding to the spherical subdivision mesh C^m.

Step 6: **Sampling.** After obtaining C^m, we find the corresponding spatial point in original surface M_Ψ for each vertex of C^m. The resulting mesh is S_Ψ^m.

Step 7: **Reconstructing boundaries.** After deleting all the vertices and triangles inside the boundary regions from S_Ψ^m, we obtain the remesh S^m.

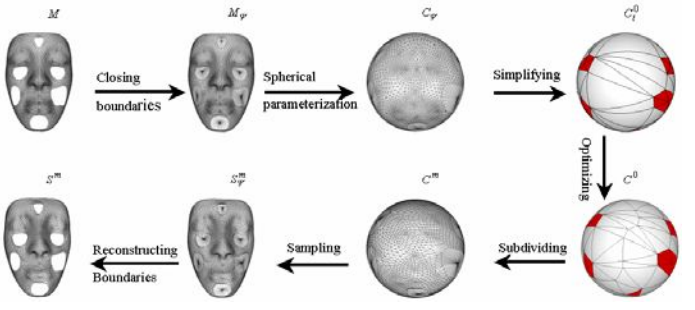


Fig.1. Frame of our remeshing method

2.2 Closing Boundaries

Since Praun's method [3] is only adapted to genus-0 meshes, all the boundaries of M have to be closed before parameterizing. In order to convenience further subdividing operation, we close the boundaries by a new strategy instead of traditional triangulation method. Fig.2 illustrates the new strategy, which inserts one new vertex v_B inside the region decided by a boundary that consists of a set of vertices $B = \{B_0, B_1, \hbar, B_n = B_0\}$. Here v_B is called one BIV (Boundary Interior Vertex) of M.

While inserting v_B, we have to find an appropriate spatial position for v_B to prevent triangles from overlapping each other. Since it is impossible to develop a universal method to decide v_B for the boundaries with arbitrary shape, we simply specify v_B to the average value of positions of all the boundary vertices. Then we scan each boundary edge $e(B_i, B_j)$ and construct a triangle $f(v_B, B_i, B_j)$ by anticlockwise order. After all boundary edges have been visited, we examine if some triangles of $T(v_B)$ overlap each other. If yes, we relocate the spatial position of v_B. Since the boundary number of a mesh is generally small, the users can complete the relocation operation manually.

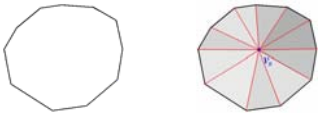


Fig. 2. Our triangulation strategy. (a) Open boundary. (b) Close boundary

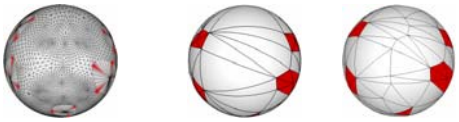


Fig. 3. Constructing process of the spherical base mesh

2.3 Constructing the Spherical Base Mesh

After mapping the closed mesh M_Ψ onto the unit sphere by Praun's method [3], we obtain the spherical parameterization mesh C_Ψ with the same connectivity as M_Ψ.

We start the construction of the base mesh C^0 by marking some special vertices of C_Ψ, which will be kept undeleted during the following mesh simplification. Then Gardland's half-edge contraction method [4] is used iteratively on C_Ψ to generate the initial base mesh C_t^0 with only marked vertices (Fig 3). Since Gardland's method selects the contraction edge by Quadric Error Metric, which doesn't consider how to optimize the triangle size, C_t^0 should be further optimized so as to generate a better base mesh C^0 (Fig 3-(c)).

2.4 Subdividing

After obtaining C^0, we iteratively execute subdividing operation on C^0 to generate the spherical subdivision mesh C^m. The subdivision level m should satisfy the inequality $H(S_\Psi^m, M_\Psi) \le \varepsilon$, where ε is the user specified threshold and $H(S_\Psi^m, M_\Psi)$ is the Hausdorff distance between the remesh S_Ψ^m and the original mesh M_Ψ (cf. Fig.1). Since S_Ψ^m is generated by using C^m to sample C_Ψ and M_Ψ, we use a vertex relocation operation to adapt the vertex distribution of C^m to that of C_Ψ, which will improve the visual appearance of S_Ψ^m.

2.5 Sampling the Original Surface

After obtaining the spherical subdivision mesh C^m, we need to find the corresponding spatial point in original surface M_Ψ for each vertex of C^m. This procedure is named as sampling the original surface. And the resulting mesh is S_Ψ^m. In this paper, we use the barycentric coordinates method to compute the corresponding spatial positions of vertices in C^m. After replacing each vertex of C^m by the corresponding spatial point, we obtain the spatial mesh S_Ψ^m. Then the remesh S^m is generated by deleting all BIVs and their 1-ring neighbor triangles from S_Ψ^m.

3 Experimental Results

We have implemented the remeshing approach and applied it to several triangle models with different number of boundaries. The original meshes are mapped onto the unit sphere by Praun's method. Fig 4 shows the remeshes of the Mask model (8,288 triangles, 7 boundaries) and the Bunny model (69,630 triangles, 2 boundaries).

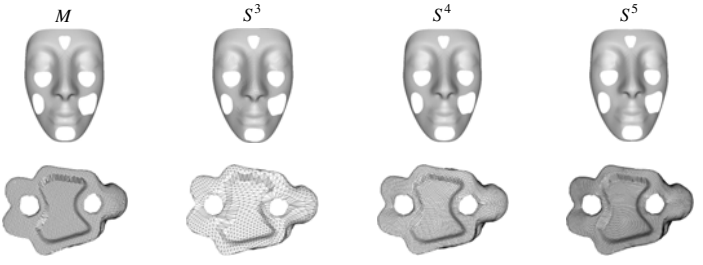


Fig. 4. The remeshing process of two different models

4 Conclusion

We have proposed an algorithm for remeshing triangle meshes with arbitrary number of boundaries. The experimental results show that our method can not only make the number of irregular vertices in the remesh as small as possible, but also preserve the boundaries of the original mesh well.

References

1. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In ACM Computer Graphics (SIGGRAPH '95 Proceedings), pages 173–182, 1995.
2. A. Lee,W. Sweldens, P. Schr¨oder, L. Coswar, and D. Dobkin. Multirelosution Adaptive Parameterization of Surfaces. In ACM Computer Graphics (SIGGRAPH '98 Proceedings), pages 95–104, 1998.
3. Praun, E., and Hoppe, H. Spherical Parameterization and Remeshing. In ACM Computer Graphics (SIGGRAPH '03 Proceedings), pages 340-349, 2003.
4. M. Garland and PS Heckbert. Surface Simplification Using Quadric Error Metrics. In ACM Computer Graphics (SIGGRAPH '97 Proceedings), pages 209-216, 1997.

SACARI: An Immersive Remote Driving Interface for Autonomous Vehicles

Antoine Tarault, Patrick Bourdot, and Jean-Marc Vézien

LIMSI-CNRS, Bâtiments 508 et 502bis, Université de Paris-Sud,
91403 Orsay, France
{tarault,bourdot,vezien}@limsi.fr
http://www.limsi.fr/venise/

Abstract. Designing a remote driving interface is a really complex problem. Numerous steps must be validated and prepared for the interface to be robust, efficient, and easy to use. We have designed different parts of this interface: the architecture of the remote driving, the mixed reality rendering part, and a simulator to test the interface. The remote driving interface is called SACARI (Supervision of an Autonomous Car by an Augmented Reality Interface) and is mainly working with an autonomous car developed by the IEF lab.

1 Introduction

The aim of the project is to develop a Mixed Reality system for the driving assistance of an autonomous car. The applications of such a system are mainly teleoperation and management of vehicles fleet. To realize such an application, we have an immersive device called MUSE (Multi-User Stereoscopic Environment) (see Fig. 1), in the LIMSI-CNRS, and an autonomous car, PiCar [1], developed by the IEF lab.

Two major concepts were used in this project: telerobotics, and telepresence. Telerobotics is a form of teleoperation in which a human acts in an intermittent way with the robot [2]. He communicates information (on goals, plans…) and receives others (on realizations, difficulties, sensor data…). The aim of telepresence is to catch enough information on the robot and its environment to be communicated to the human operator in such a way that he should feel physically present on the site [3].

We took two existing interfaces as a starting point. In [4], Fong and al. define a collaborative control between the vehicle and the user. Queries are send to the robot, which executes them or not, depending on the situation. The robot can also send queries to the user, who can take them into account. This system can be adapted to the level of expertise of the user. The depth parameter of the scene is given by a multisensor fusion of a ladar, monochrome camera, a stereovision system, an ultrasonic sonar, and an odometer. They have developed two interesting driving interface: "gesture driver", allows the user to control the vehicle with a series of gesture. Unfortunately, this driving method is too tiring for long distances. PDAdriver, enables to drive a robot with a PDA. In [5], McGreevy describes a virtual reality interface for efficient remote driving. His goal was to create an explorer-environment interface instead of a classical computer-user interface. All the operations, objects, and contexts must be comparable to those met in a natural environment.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 339 – 342, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Fig. 1. MUSE immersive device

2 Global Architecture of the System

To answer the time and situational consciousness constraints of a telepresence system, we had to prefer the transmission of video data from PiCar to a virtual reconstruction of its environment. SACARI must also send/receive data for/from the wheel orientation and speed actuators/sensors. We transmit video and orders on two separate channels. PiCar controlling station is connected to the immersive device by a gigabit network between the IEF lab and the LIMSI-CNRS (see Fig. 2).

Fig. 2. Architecture of the system

To fulfill the telepresence constraints, we chose to make a system with short time delay, distributed operations, and modular components. We also needed software for data exchange, and the possibility to record/replay data from different tests. SACARI and PiCar's software part are developed on the same platform addressing these constraints: RTMaps (http://www.intempora.com). Each module is represented as a *Component*. We have developed three main groups of components for our application: a devices driver component, a rendering component, and a vehicle behavior simulator.

3 A Scene Graph-Based Renderer

The rendering Components have several constraints. First, it has to be able to simulate PiCar's environment. It should integrate video texture and complex virtual objects. The renderer library must be easily changed and inserted in a multithreaded application (RTMaps). Finally, the system should support multiscreen display, as our immersive device is biplan. The use of two screens is primordial. The telepresence recommends that the user feels present on site. That's why most of his field of view must be filled by the virtual scene. Moreover, we have noticed that a single screen limits the

range of action of a user, especially when he wants to change the vehicle's direction using a 6 DOF tracker: he tends to give a direction only lying in the vision range. These constraints made us choose OpenSceneGraph (www.openscenegraph.org).

We have developed several kinds of "OSG Components" in RTMaps: Transformations, graphical objects which represent the leaves of the scene graph, video objects that can read RTMaps-produced image flow, and a viewer. This component can be set to drive a cluster for graphical rendering, switch between different kinds of navigation, specify graphical and stereo settings.

4 The Car Simulator

The device allowing the user to control the vehicle should work in semi-autonomous *and* non-autonomous modes. That's why we chose to use a 6 DOF Tracker to manage vehicle remote driving instead of a classical steering wheel. To drive the tracker, we use our own devices drivers manager: VEserver [6]. It is a real time client/server devices drivers manager with a distributed architecture that can drive synchronously numerous devices. We have integrated a client of the VEserver in RTMaps. An extern VEserver node tracks the events coming from an ARTTrack wireless 6DOF tracker. The vehicle is driven as presented in Fig. 3.

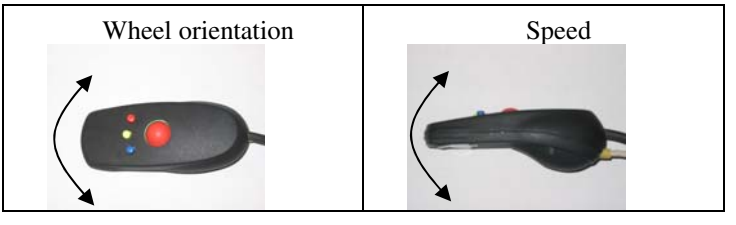


Fig. 3. Using a 6 DOF Tracker to drive the vehicle

We developed a navigator transforming the speed and the wheel orientation, given by the 6 DOF tracker, into a position and orientation of the scene graph camera. Given the last camera orientation ψ, the speed v and the front steering β, the back steering α and the length L between the nose gear wheels and the aft wheels, we can calculate the differential position and orientation of the vehicle:

$$\dot{x} = v \cdot \cos\psi \qquad \dot{y} = v \cdot \sin\psi$$

$$\dot{\psi} = \frac{v}{l}\tan\beta \qquad \text{with} \quad l = \frac{\alpha L}{1+\alpha}$$

Then, we integrated the autonomous car simulator, realized by the IEF lab, into our system. The component we developed takes the wanted position and orientation in input and gives the trajectory, the speed and the wheel orientation on output. The display shows the calculated trajectory and the next wanted point to reach (see Fig. 4).

Fig. 4. Control of the autonomous vehicle

5 Conclusion and Perspectives

We have developed all the needed tools for a remote driving application:

- an easy-to-use scene graph descriptor, which can be reused for VR applications,
- A simulator to test the different ways to control PiCar
- An interface dedicated to the remote driving and supervision of the vehicle

The next step will be to test our interface in real conditions, for remote driving and supervision. Another awkward point will be the transition from supervision to remote driving of the vehicle. It must be as natural as possible for the user. We plan to test different devices to perform such a transition.

References

1. S. Bouaziz, M. Fan, A. Lambert, T. Maurin, R. Reynaud, "PICAR: experimental Platform for road tracking Applications", *IEEE IV2003 Intelligent Vehicle Symposium,* ISBN 0-7803-7848-2*,* pp. 495-499, Columbus, June 2003
2. H. S. Tan, K. H. Lee, K. K. Kwong, "VR Telerobot System", *Proceedings of the 5th International Conference on Manufacturing technology,* Beijing, November 1999
3. B. Hine, P. Hontalas, T. Fong, L. Piguet, E. Nygren, "VEVI : A Virtual Environment Teleoperations Interface for Planetary Exploration", *SAE 25th International Conference on Environmental Systems*, San Diego, July 1995
4. T. Fong, C. Thorpe, C. Baur, "Advanced interfaces for Vehicle Teleoperation: Collaborative Control, Sensor Fusion Displays, and Remote Driving Tools", *Autonomous Robots 11*, pp. 77-85, 2001
5. M. W. McGreevy, "Virtual Reality and Planetary Exploration", *A. Wexelblat (Ed.), Virtual Reality: Application and Explorations*, pp. 163-167, 1993
6. D. Touraine, P. Bourdot, Y. Bellick, L. Bolot, "A framework to manage multimodal fusion of events for advanced interactions within Virtual Environments", *8th Eurographics Workshop on Virtual Environment*, 2002

A 3D Model Retrieval Method Using 2D Freehand Sketches

Jiantao Pu and Karthik Ramani

Purdue Research and Education Center for Information Systems in Engineering (PRECISE), Purdue University, West Lafayette IN 47907-2024
{pjiantao, ramani}@purdue.edu

Abstract. In this paper, a method is proposed to retrieve desired 3D models by measuring the similarity between a user's freehand sketches and 2D orthogonal views generated from 3D models. The proposed method contains three parts: (1) pose determination of a 3D model; (2) 2D orthogonal view generation along the orientations; and (3) similarity measurement between a user's sketches and the 2D views. Users can submit one, two or three views intuitively as a query, which are similar to the three main views in engineering drawing. It is worth pointing point out that our method only needs three views, while 13 views is the minimum set that has been reported by other researchers.

1 Introduction

Up to this point, many methods have been proposed to retrieve the desired models from a database. These methods can be classified into four categories: feature-vector based method [1]; statistics-based method [2]; topology-based method [3]; and image-based method [4]. An advantage of the feature-based method is their simplicity, but there is no feature or feature set that can describe all 3D shapes in a uniform way. The statistics-based methods do not require pose registration and feature correspondence, and are fast and easy to implement, but they are not sufficient to distinguish similar classes of objects. From the perspective of structural similarity, topology-based methods have many desired properties, such as intuitiveness, invariance, and robustness. And not only global features but also local features are depicted. However, they require a consistent representation of an object's boundary and interior, and it is not easy to register two graphs robustly. The motivation of the image-based method is to imitate the ability of the human visual system to recognize objects. However, many images from different perspectives are needed.

In this paper, we propose a method to retrieve 3D models by measuring the similarity between a user's sketches and three 2D orthogonal views generated from 3D models. The idea arises from a practice: *engineers usually express their concept of a 3D shape with three 2D views without missing any information*. For this purpose, we present three algorithms: (1) pose normalization of 3D objects; (2) 2D drawing-like view generation; and (3) similarity measurement between 2D views. In the remainder of this paper, the approaches to the three problems will be described respectively, and some experimental results are presented to show its validity.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 343 – 346, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Pose Normalization

As a classical statistical method, principal component analysis (PCA) [5] is used to estimate the intuitive directions along which the mass is heavily distributed. However, it is not good enough at aligning orientations of different models within similar shapes. Extended Gaussian Image (EGI) [6] is another classical method to determine the pose of a 3D object. For a 3D object, each point on its Gaussian sphere corresponds to a particular surface orientation and the respective surface area. However, for nonconvex objects, different shapes may have the same EGI representation. To overcome the above-mentioned limitations, we propose a new orientation determination method. A 3D shape is represented by a triple $\mathbf{S}=\{\ p_i \mid (\mathbf{N}_i, A_i, D_i),\ 0 \le i \le n\ \}$, in which $\mathbf{N}_i$ represents the normal of polygon p_i, A_i, represents the area, and D_i represents the distance between the mass center C and the polygon p_i. Our aim is to find out the normal along which the summed surface area is the largest and these surfaces have the same distance from the mass center:

Step 1: Compute the normal direction $\mathbf{N}^k$ for each triangle $\Delta\,\mathbf{p}^k\mathbf{q}^k\mathbf{r}^k$ and normalize it. The normal of a triangle is equal to the cross product of its two edges:

$$\mathbf{N}^k = \frac{\mathbf{p}^k\mathbf{q}^k \times \mathbf{q}^k\mathbf{r}^k}{\left\|\mathbf{p}^k\mathbf{q}^k \times \mathbf{q}^k\mathbf{r}^k\right\|} \tag{1}$$

Step 2: Summarize the areas of all triangles with the same normals and same distance from the center mass.

Step 3: Determine the three principal axes. The normal associated with the maximum areas is selected as the first principal axis $\mathbf{b}^u$. To get the next principal axis $\mathbf{b}^v$, we can search from the remaining normal distributions and find out the normal that satisfies two conditions: (a) with maximum areas; and (b) orthogonal to the first normal. The third axis can be obtained by doing cross product between $\mathbf{b}^u$ and $\mathbf{b}^v$.

Step 4: Find the center and the half-length of the bounding box. This can be done by projecting the points of the convex hull onto the direction vector and finding the minimum and maximum along each direction. In Figure 1, some examples obtained by the MND method are shown.

Fig. 1. Some pose determination examples

3 2D Orthogonal View Generation

To compute the view of a 3D model on a plane, we design an algorithm and explain it as follows with the help of the example shown in Figure 2.

Step 1: Backface culling in object space. When engineers represent a 3D object using 2D views, the invisible backfaces are not considered. Given a projection direction **n** and a polygon P*i* with normal vector $\mathbf{n}_i$, if $\mathbf{n}_i \cdot \mathbf{n} > 0$, then this polygon is visible; otherwise, it is invisible. Figure 2(b) shows the backface culling result for the model shown in Figure 2(a).

Step 2: Inside-edge culling. To obtain the drawing-like view of 3D objects, the inside-edges have to be discarded. The inside-edge has a distinguishing property: *it is shared by two polygons*. With this definition, we can cull the inside-edges completely by transcending all the triangles. The result is shown in Figure 2(c).

Step 3: Orthogonal projection along the viewing direction, as Figure 2(d) shows. An example obtained by this method is shown in Figure 3.

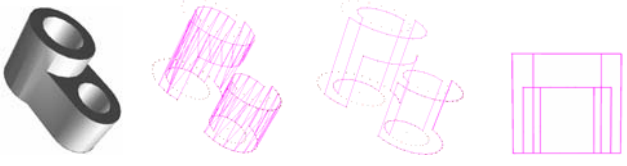


Fig. 2. Four steps for view generation: (a) a 3D model; (b) the result after backface culling; (c) the result after inside-edge culling; and (d) the generated 2D view

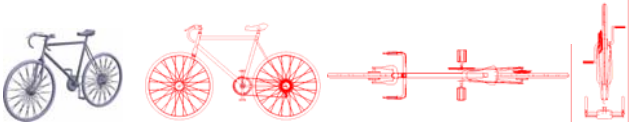


Fig. 3. A 2D view generation example

4 2D Shape Distribution Method for Similarity Measurement

To measure the similarity between 2D views, we propose a 2D shape histogram method to measure the similarity between 2D views:

Step 1: Uniform sampling on view edges. From the statistics perspective, a 2D shape can be approximated by enough sampled points. This process can be done by an unbiased sampling strategy similar to the method adopted in [7].

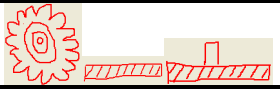
Step 2: Shape distribution generation. By summarizing the numbers of point pairs with the same distance, the 2D shape distribution can be generated.

Step 3: Similarity measurement. Generally, there are two normalization methods to account for the size difference between two views: (a) align the maximum D2 distance values, and (b) align the average D2 distance values. The similarity between the two views is measured by calculating the difference between their distributions in the form of a histogram.

5 Experimental Results

An intuitive application for this proposed method is sketch-based user interface, in which the query process is similar to when engineers represent a 3D shape on a piece of paper. In order to evaluate its validity and performance, some experiments have been conducted and show that our proposed method has many valuable advantages: (a) insensitive to geometric noise; (b) invariant to translation, rotation, and scaling; and (c) supportive of freehand sketch query. In Table 1, two retrieval examples using freehand sketches are presented:

Table 1. Two retrieval examples using freehand sketches

Sketches	Top Four Similar Models			
				

6 Conclusion

This paper presents a 3D model retrieval method by measuring the similarity between 2D views. The method enables the intuitive implementation of a 2D sketch user interface for 3D model retrieval. In the future, we will focus our attention on local shape matching, in which users can specify some local shape explicitly

References

1. Elad, M., Tal, A., Ar, S.: Content Based Retrieval of VRML Objects: An Iterative and Interactive Approach. Proc. 6th Eurographics Workshop on Multimedia 2001, Manchester, UK (2001) 107–118.
2. Paquet, E., Rioux, M.: Nefertiti: A Tool for 3-D Shape Databases Management. SAE Transactions: Journal of Aerospace 108 (2000) 387–393.
3. Hilaga, M., Shinaagagawa, Y., Kohmura, T., Kunii, T.L.: Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. Proc. SIGGRAPH 2001, Los Angeles, USA, (2001) 203–212.
4. Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., Jacobs, D.: A Search Engine for 3D Models. ACM Transactions on Graphics, Vol.22 (1): 83–105 (2003).
5. Petrou, M., and Bosdogianni, P., "Image Processing: The Fundamentals," John Wiley (1999).
6. Horn, B.K.P., "Extended Gaussian Images," Proc. IEEE, Vol.72 (12): 1671–1686 (1984).
7. Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D.: Shape Distribution. ACM Transactions on Graphics, Vol.21 (4): 807–832 (2002).

A 3D User Interface for Visualizing Neuron Location in Invertebrate Ganglia

Jason A. Pamplin[1], Ying Zhu[1], Paul S. Katz[2], and Rajshekhar Sunderraman[1]

Departments of [1]Computer Science and [2]Biology,
Georgia State University,
Atlanta, Georgia, USA
jason@internetworkflow.com
{yzhu, pkatz, rsunderraman}@gsu.edu

Abstract. Invertebrate nervous systems serve as important models for neuroscience research because they are comprised of relatively small numbers of individually identified neurons. There is no universal means of documenting locations of individual neurons that allows variability between specimens and can be generalized to different species. We present a new technique for visualizing and documenting neuron location. First, we describe a 3D user interface that allows neuroscientists to directly mark neuron locations on a 3D deformable model. A new mapping scheme is proposed that specifies the location of a neuron in a common coordinate system that accommodates the individual variability in size and shape of ganglia.

1 Introduction

Understanding brain function depends upon identifying neuronal elements and their connections. Molluscan nervous systems have provided important models in studies of learning, memory [1] and motor pattern generation[2] because they are comprised of individually identifiable neurons. The brains of these animals contain about 6000-10,000 neurons clustered in ganglia. In contrast, the mammalian brain has about 10^{11} neurons, which fall into about 6000 classes. Therefore, the molluscan nervous system can be used as a model for developing a database of neurons and connections if the model includes a method of identifying and recording the location of each neuron's cell body.

In opisthobranch molluscs, such as *Tritonia diomedea*, which we are using as our model system, the cell bodies of neurons lie on or near the surface of the ganglia. Two mapping problems must be solved: (1) a 3-D UI must allow for individual shape and size variations of *Tritonia* ganglia so neuroscientists can mark the locations of a neuron; and (2) coordinates must be transformed to a common coordinate system, independent of the specimen geometry, such that location information can be searchable.

2 Background

Neuron localization is the process of assigning each neuron a coordinate so that one can recognize the same or similar neurons in different brain specimens. Neuron local-

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 347 – 350, 2005.
© Springer-Verlag Berlin Heidelberg 2005

ization is difficult because (1) brains can be variable in shape; (2) different terms are used to describe a given brain region in different species, and the same term is used to describe different regions; and (3) boundaries of regions are sometimes ambiguously defined [3].

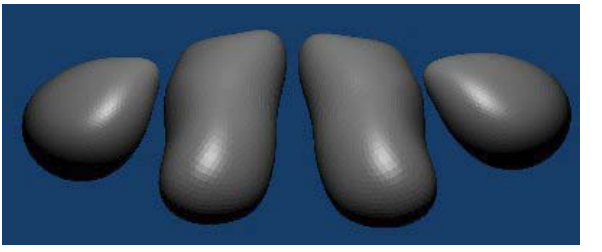


Fig. 1. Simple 3D model of Tritonia brain

Many research efforts have addressed the neuron localization problem [4][5][6][7]. The resulting method can be summarized as follows: (1) A 2D or 3D brain atlas is created from brain cross section images. The most notable example is the Talairach-Tournoux brain atlas [8]. (2) A coordinate system is defined based on certain features or landmarks on the brain atlas. (3) The atlas is then manipulated to match the features or landmarks on the given target dataset, or vice versa. The manipulations range from simple linear transformations to sophisticated physics based deformations[9][10]. (4) After the atlas is fitted to the data set, the features are assigned coordinates. This method, which depends on brain cross-section images, is used when important brain structures reside on the interior of the brain. In molluscan brains, however, neurons reside at or near the surface, allowing for a simpler approach.

3 3D User Interface for Neuron Location Identification

The user interface, while being easy to use, should also accurately represent neuron location relative to a brain atlas. The brain atlas is created using a standard 3D graphics package -- Blender. Figure 1 shows a reference 3D model composed of 4 individual lobes that are generated separately and then placed together to form a complete model of the *Tritonia* brain. Figure 2 shows a photo of a typical *Tritonia* brain.

The reference model must be adjusted (deformed) to visually match the actual specimen. To help the user accurately deform the model, our interface will display a photo of the specimen, overlaid with the semi-transparent 3D brain atlas. The user deforms the atlas with global scaling, rotation, and translation, to approximate the photo. Then, local deformations can fine-tune the match. A "Wire" deformation [11] algorithm is adopted here, due to its simplicity, efficiency and good interactive control. To use the atlas, the user marks a point that matches the

Fig. 2. Photo of typical Tritonia brain Scale bar is 0.5mm

location of the neuron under study. The location information can then be extracted by the system and stored in the database. The interface also can be used in reverse, i.e. upon selecting a location or a small region, information about neuron(s) at that location or within the region can be retrieved from the database and displayed to the user.

4 A Common Coordinate Space for Neuron Localization

An obvious solution to neuron localization is to keep a unique index of each vertex, record the index of the selected vertex, and then retrieve the reference x, y, z coordinates from an index table. There are several problems with this approach: (1) excessive memory is required to store an index to reference vertex table; (2) changes in the reference model that result in re-indexing will corrupt location information; and (3) storage of the reference x, y, z will make marking of a deformed model computationally intensive.

We propose mapping the 3D atlas to a 2D image space with texture mapping. Each vertex on the 3D atlas will have a corresponding texture coordinate, which is calculated with a standard parametric equation when the texture is applied to the model. Thus, each vertex on the 3D atlas is mapped to a pixel on the texture image. Once the initial correspondence is established, the texture coordinates are not recalculated during the deformation process described in section 3. Thus, for *Tritonia* neuron localization, the texture coordinate is stored instead of the x, y, z coordinates in 3D space.

The size and shape of the brain atlas may change, but the texture mapping remains stable, since the same parametric equation was used to map the texture coordinates for the vertices. As long as the texture is mapped completely onto the object, the relative location of each pixel of the texture image will be in the same location within a small margin of error. Thus the texture image provides a common coordinate space for comparing neurons. This technique allows for the calculation of average location for a series of observations and a statistical view of neuron location.

5 Results and Discussion

We have created a prototype *Tritonia* brain model (Figure 1) using an open source modeling tool. We are designing and developing a simple interface that allows users to mark neuron locations. The proposed solution has the following benefits: (1) changes to the reference model will not alter the location information, even if the reference vertex locations change; (2) texture coordinate calculation is performed by the modeling software with minimal computing cost; and (3) our neuron localization algorithm can be easily adapted to other species by simply creating a new 3D model.

Our solution also has it limitations: (1) this method works only for brain models where neurons are on or close to the surface; (2) users may need practice to match the atlas model to a specimen; and (3) the texture mapping may not be an exact one-to-one mapping, which may lead to some inaccuracy in neuron mapping – this problem can be reduced by matching the resolution of the 2D image with that of the 3D atlas.

6 Conclusion and Future Work

We have discussed our 3D user interface for marking neuron locations directly onto a 3D brain atlas. The 3D interface and the deformable 3D brain atlas can also be used to query the neuron database using positional information. We also discussed a new method for mapping neuron locations by using a technique similar to texture mapping. As a result, the user is able to compare the same neuron on different brain specimens in a common coordinate system. Although our algorithm is primarily designed for *Tritonia* brains, it can be easily adapted to other species by introducing a new brain atlas and adjusting the mapping equations.

Acknowledgments

Many thanks to Georgia State University colleagues: Robert Calin-Jageman, Jim Newcomb, Hao Tian, Christopher Gardner, and Lei Li; also thanks to the participants of the Identified Neuron Database Workshop (Georgia State University, December 2004). This research is funded in part by a Georgia State University Brains & Behavior Program Seed Grant and a Georgia State University Research Initiation Grant.

References

1. Pittenger, C., Kandel, E.R., In Search of General Mechanisms for Long-lasting Plasticity: Aplysia and the hippocampus. Philosophical Transactions of Royal Society of London, Series B: Biological Science. 358(1432), (2003) 757-63
2. Getting, P.A., A Network Oscillator Underlying Swimming in Tritonia. Neuronal and Cellular Oscillators. J.W. Jacklet, Editor. Marcel Dekker, Inc., New York. (1989) 215-236
3. Bjaalie, J.G., Localization in the Brain: New Solutions Emerging. Nature Reviews: Neuroscience. 3 (2002) 322-325
4. Davatzikos, C., Spatial Normalization of 3D Brain Images Using Deformable Models. Journal of Computer Assisted Tomography, 20(4), (1996) 656-65
5. Gee J.C., Reivich, M., Bajcsy R., Elastically Deforming an Atlas to Match Anatomical Brain Images. Journal of Computer Assisted Tomography, 17(2), (2003) 225-236.
6. Roland, P.E., Zilles, K., Brain Atlases - a New Research Tool. Trends in Neuroscience, 17(11), (1994) 458-67
7. Payne B.A., Toga A.W., Surface Mapping Brain Function on 3D Models. IEEE Computer Graphics Applications, 10(5), (1990) 33-41.
8. Talairach, J., Tournoux, P., Co-planar Stereotaxic Atlas of the Human Brain. Thieme Medical Publishers, New York. (1988)
9. Toga, A.W., Brain Warping. Academic Press, New York. (1998)
10. Thompson, P., Toga, A.W. A Surface-based Technique for Warping Three-dimensional Images of the Brain. IEEE Transactions on Medical Imaging, 15(4), (1996) 402-417
11. Singh, K., Fiume, E., Wires: A Geometric Deformation Technique. Proceedings of the 25th ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), (1998) 405-414

The Dynamics of General Fuzzy Cellular Automata[1]

Angelo B. Mingarelli

School of Mathematics and Statistics, Carleton University,
Ottawa, Ontario, Canada, K1S 5B6
amingare@math.carleton.ca

Abstract. We continue the investigation into the dynamics and evolution of fuzzy rules, obtained by the fuzzification of the disjunctive normal form, and initiated for rule 90 in [2], for rule 110 in [10] and for rule 30 in [3]. We present general methods for detecting the evolution and dynamics of any one of the 255 fuzzy rules and apply this theory to fuzzy rules 30, 110, 18, 45, and 184, each of which has a Boolean counterpart with interesting features. Finally, it is deduced that (except for at most nine cases) no fuzzy cellular automaton admits chaotic behavior in the sense that no sensitive dependence on the initial string can occur.

1 Introduction

This work is motivated by a question originally posed by Andy Wuensche [1] regarding the convergence of fuzzy rules induced by fuzzy cellular automata (CA). He asked whether the results in [2] for fuzzy rule 90 presented in the conference cited in [1] could be extended to fuzzy rule 30 in the generality obtained in [2]. Although this was answered in [3] we choose to go beyond this and provide a framework for discovering the global evolution of an arbitrary fuzzy CA, cf., [4]. We develop some methods for obtaining limiting information about any one of the 255 fuzzy rules. Recent work in this new area includes some variations on the game of Life in [5] and applications to pattern recognition [6]. In addition, such CA have been used to investigate the result of perturbations, for example, noisy sources, computation errors, mutations, etc. on the evolution of boolean CA (cf., [7], [4], [2], etc.).

Recall some basic terminology from [2]. A CA is a collection of cells arranged on a graph; all cells share the same local space, the same neighborhood structure and the same local function (i.e., the function defining the effect of neighbors on each cell). Given a linear bi-infinite lattice of cells, the local Boolean space $\{0, 1\}$, the neighborhood structure (left neighbor, itself, right neighbor), and a

[1] This research is partially supported by an NSERC Canada Discovery Grant and by a grant from the Office of the Vice-President Research and International, Carleton University.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 351–359, 2005.
© Springer-Verlag Berlin Heidelberg 2005

local rule $g:\{0,1\}^3 \to \{0,1\}$, the global dynamics of an *elementary CA* are defined by (cf., [8]) $f:\{0,1\}^{\mathbb{Z}} \to \{0,1\}^{\mathbb{Z}}$ and $f(x)_i = g(x_{i-1}, x_i, x_{i+1})$, for all i. The *local rule* is defined by the 8 possible local configurations that a cell detects in its neighborhood: $(000, 001, 010, 011, 100, 101, 110, 111) \to (r_0, \cdots, r_7)$, where each triplet represents a local configuration of the left neighbor, the cell itself, and the right neighbor. In general, the value $\sum_{i=0}^{7} 2^i r_i$ is used as the *name* of the rule. As usual, the local rule of any Boolean CA is expressed as a *disjunctive normal form*: $g(x_1, x_2, x_3) = \vee_{i|r_i=1} \wedge_{j=1}^{3} x_j^{d_{ij}}$ where d_{ij} is the j-th digit, from left to right, of the binary expression of i, and x^0 (resp. x^1) stands for $\neg x$ (resp. x). A *Fuzzy CA* is obtained by *fuzzification* of the local function of a Boolean CA: in the disjunctive normal form by redefining $(a \vee b)$ as $(a+b)$, $(a \wedge b)$ as (ab), and $(\neg a)$ as $(1-a)$. The usual fuzzification of the expression $a \vee b$ is $max\{1, a+b\}$ so as to ensure that the result is not larger than 1. Note, however, that taking $(a+b)$ for the CA fuzzification does not lead to values greater than 1 since the sum of all the expressions for rule 255 is 1 (*i.e.,* $g_{255}(x, y, z) = 1$), and so every (necessarily non-negative) partial sum must be bounded by 1. Since every fuzzy rule is obtained by adding one or more of these partial sums it follows that every fuzzy rule is bounded below by 0 and above by 1. We will be analyzing the behavior of an *odd* fuzzy rule, rule 45, towards the end of this paper.

As an example, we note that rule $45 = 2^0 + 2^2 + 2^3 + 2^5$ has the local rule $(000, 001, 010, 011, 100, 101, 110, 111) \to (1, 0, 1, 1, 0, 1, 0, 0)$. Its canonical expression is $g_{45}(x_1, x_2, x_3) = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3)$ and its fuzzification gives $g_{45}(x_1, x_2, x_3) = 1 - x_1 - x_3 + x_2 x_3 + 2x_1 x_3 - 2x_1 x_2 x_3$. In the same way we derive the local rules for rules 45 and 184. One of the exceptional rules (one of nine that defies assumption (I), stated in the next section), is rule 184 (see [4]) which we will analyze at the very end. The dynamics of these nine rules are interesting in that the methods presented herein require some modification, yet even so, it cannot be asserted at this time that we can determine their dynamics in general. See the last subsection for details.

Let $g_n(x_1, x_2, x_3)$, $1 \le n \le 255$, denote the canonical expression of fuzzy rule n. We know that the disjunctive normal form for a fuzzy rule is given by $g_n(x_1, x_2, x_3) = \vee_{i|r_i=1} \wedge_{j=1}^{3} x_j^{d_{ij}}$ where $0 \le d_{ij} \le 1$ is the integer defined above. Since $x_j^0 = 1 - x_j$, $x_j^1 = x_j$, and the disjunction is an additive operation it follows that g_n is a linear map in each variable separately and so satisfies Laplace's equation (see [9]). Thus, maximum principles (see [9], Chapter 4) can be used to derive properties of such rules under iterations.

2 The Long Term Dynamics of General Rules

We fix the notation and recall definitions from [2]. The *light cone* from a cell x_i^t is the set of $\{x_j^{t+p} \mid p \ge 0 \,\text{and}\, j \in \{i-p, \cdots, i+p\}\}$. In this case, the light cone is the boundary of an infinite triangle whose vertex is at the singleton a and whose boundary consists of all the other a's. Thus, $x_{\pm n}^m$ will denote the cell at $\pm n$ steps to the right/left of the zero state at time m. The single cell x_0^0 will

be denoted by a and generally we will take it that $0 < a \leq 1$, since $a = 0$ is clear. The method we present will allow us to determine the long term dynamics of any fuzzy rule $g_n(x, y, z)$ where $1 \leq n \leq 255$ where the various asymptotic estimates are found via successive iterations. In a nutshell, the basic idea here is to distinguish a single diagonal for a starting point, use the rule to derive basic theoretical estimates, use continuity to prove the existence of the various limits, when applicable, and finally use an iterative scheme to compute all subsequent limits.

We will always assume that $g : [0, 1]^3 \to [0, 1]$ is continuous on the unit cube $\mathcal{U}$ and not necessarily the canonical expression of a fuzzy rule. This is for simplicity only since, in reality, any compact set may be used in lieu of $[0, 1]$. The symbol $G^m(a)$ denotes the usual m^{th} iterate of G at a, where $G(a) \equiv g(0, 0, a)$ and $a \in (0, 1)$ is a given fixed number, sometimes called a *seed*. Similarly, we define $H(a) \equiv g(a, 0, 0)$. The value of "a" here measures in some sense the degree of fuzziness in that $a = 0$ gives trivial evolutions while $a = 1$ gives Boolean evolution. We assume, as a further restriction on g, that

(I) The equations $x - g(x, y, z) = 0$, $y - g(x, y, z) = 0$, and $z - g(x, y, z) = 0$ may each be solved uniquely for x, y, z respectively, for given values of (y, z), (x, z), (x, y) respectively in $[0, 1]^2$, and that the resulting functions of (y, z), (x, z), (x, y) are continuous on $[0, 1]^2$.

(II) The limits $G^m(a) \to L_0^-(a)$ and $H^m(a) \to L_0^+(a)$ each exist and are finite as $m \to \infty$.

Remark. Condition (I) is generally satisfied for fuzzy rules as considered here. The only exceptions that require modifications to the technique presented here are the nine fuzzy rules 170, 172, 184, 202, 204, 216, 226, 228, and 240. They are to be distinguished because they violate (I). In general, the implicit function theorem may be used here to guarantee conditions under which (I) holds, for arbitrary local rules. Secondly, we note that the full force of hypothesis (I) is not necessary for many of the fuzzy rules and that weaker assumptions can be made by restricting the class of rules. Nevertheless, we will assume it throughout for expository reasons and introduce modifications as the need arises. For most fuzzy rules (II) clearly holds because of our basic assumptions. For odd rules this assumption (II) may fail, but the techniques herein can then be applied to subsequences (see fuzzy rule 45 below).

2.1 Evolution and Dynamics for a Single Seed in a Zero Background

We assume that all cells but one (denoted by a) are initially set at zero. Writing $g(0, 0, a) \equiv G(a)$, we see that the main left-diagonal $x_{-m}^m = G^m(a)$ for each $m \geq 1$, where the symbol $G^m(a)$ denotes the usual m^{th} iterate of G at a. The cells of the first left-diagonal (given by x_{-m}^{m+1}, $m \geq 0$) then satisfy $x_m^{m+1} = g\left(0, G^m(a), x_{-(m-1)}^m\right)$. Passing to the limit as $m \to \infty$ in the previous display and using (II), we see that $L_1^-(a) = g\left(0, L_0^-(a), L_1^-(a)\right)$, by (I),

and so this relation defines this limit $L_1^-(a)$ uniquely. Now that we know both $L_0^-(a)$ and $L_1^-(a)$ we can find $L_2^-(a)$ since $x_{-n}^{n+2} = g\left(x_{-(n+1)}^{n+1}, x_{-n}^{n}, x_{-(n-1)}^{n+1}\right)$, for each $n \geq 1$. Passing to the limit as $n \to \infty$ we find the special relation $L_2^-(a) = g\left(L_0^-(a), L_1^-(a), L_2^-(a)\right)$. By (I) this equation can be solved uniquely for $L_2^-(a)$, since the other two quantities are known. Proceeding inductively we observe that

$$L_{k+1}^-(a) = g\left(L_{k-1}^-(a), L_k^-(a), L_{k+1}^-(a)\right) \tag{1}$$

holds for each $k \geq 1$ and this defines the limit $L_{k+1}^-(a)$ uniquely, by (I), in terms of the preceding limits. If we set $g(a,0,0) \equiv H(a)$, then $x_m^m = H^m(a)$ for each $m \geq 1$. Arguing as in the left-diagonal case and using (II) we get that the existence of $\lim_{m\to\infty} H^m(a) = L_0^+(a)$ implies the existence of $L_1^+(a)$. The limit $L_2^+(a)$ is now found recursively as the unique solution of $L_2^+(a) = g\left(L_2^+(a), L_1^+(a), L_0^+(a)\right)$ whose existence is guaranteed by (I). Finally, an inductive argument gives us that subsequent limits, like $L_{k+1}^+(a)$, are given recursively by solving

$$L_{k+1}^+(a) = g\left(L_{k+1}^+(a), L_k^+(a), L_{k-1}^+(a)\right) \tag{2}$$

uniquely, using (I), for each $k \geq 1$.

As for the limits of the right-vertical sequences of the form $\{x_m^{m+p}\}_{p=0}^{\infty}$, $m = 1, 2, 3, \ldots$, we use the fact that the union of two sets of points each of which has exactly one (possibly the same) point of accumulation (or limit point) also has exactly two (maybe the same) points of accumulation. This result is easily derived using topology and, in fact, it also holds for any countable family of sets each of which has exactly one point of accumulation (by the Axiom of Choice). In the case under discussion we note that the right-half of the infinite cone, $\mathcal{C}^+$, whose vertex is at x_0^1 (the set that excludes the zero background and the zeroth left- and right-diagonals) can be written as $\mathcal{C}^+ = \bigcup_{i=1}^{\infty} \mathcal{S}_m^+ = \bigcup_{i=1}^{\infty} \{\{x_j^p\}_{j=0}^{\infty} |\ p \geq m\}$, and is therefore the countable union of sets of (right-diagonal) sequences each of which converges to some $L_k^+(a)$ and so the only points of accumulation of $\mathcal{C}^+$ must lie in the union of the set consisting of all the limits $L_k^+(a)$, for $k \geq 1$. Since right-vertical sequences are infinite subsequences of $\mathcal{C}^+$, we get that all such sequences may or may not have a limit but if they do, it must be one of the $L_k^+(a)$'s where the k generally depends on the choice of the column. A similar discussion applies to left-vertical sequences.

Remark. If the limits $L_k^{\pm}$ themselves converge as $k \to \infty$ to $L^{\pm}$, say, then $L^{\pm}$ must be a fixed point of the rule, that is $L^{\pm} = g\left(L^{\pm}, L^{\pm}, L^{\pm}\right)$. This is clear by (1), (2) and the continuity of g. The same observation applies to limits of vertical sequences. In the case of fuzzy rule 30 all columns (left or right) converge to one and the same value, namely, 1/2 (see [3]). On the other hand, fuzzy rule 18 (see below) has the property that, generally speaking, *none* of its vertical columns converges at all (since each column has exactly two limit points).

2.2 A General Theory in the Finite Support Case

This case is very similar to the single support case and so need only be sketched. We now assume that all cells but a finite number are initially set at zero. We take it that the initial string of cells is given by $x^0_{-k}, \ldots, x^0_0, \ldots, x^0_q$, where $x^0_{\pm i} \in (0,1)$. Without loss of generality we will assume that $k \geq 1$ and $q \geq 0$ so that our initial string contains at least two elements.

We now distinguish two vertical columns, that is, $\mathcal{V}_{-k}$ and $\mathcal{V}_q$, that is those two infinite columns whose top cell is x^0_{-k}, x^0_q, respectively. We need to describe the evolution of the *half*-light cones emanating to the left and down from x^0_{-k} (and to the right and down from x^0_q). Suppressing the variables in the expression of a limit for the purpose of clarity we will write $L^-_{-k}(x^0_{-k}, \ldots, x^0_0, \ldots, x^0_q)$ as L^-_{-k}. As before the zeroth left-diagonal, consisting of the value x^0_{-k} only, necessarily converges to the same value. Hence, $L^-_0 = x^0_{-k}$. It is helpful to think of the cell value x^0_{-k} as playing a role analogous to a in the single support case.

Consider now the first left-diagonal S^-_1, originating at the cell $x^0_{-(k-1)}$ of the initial string. By definition every term of S^-_1 on or to the left of $\mathcal{V}_{-k}$ is of the form $x^{m+1}_{-(k+m)} = g\left(0, x^0_{-k}, x^m_{-(k+m-1)}\right)$, where $m \geq 0$. Passing to the limit as $m \to \infty$, using the continuity assumptions of g at the outset, and hypotheses (I) and (II) we see that $L^-_1 = g(0, L^-_0, L^-_1)$, from which we get the existence and uniqueness of L^-_1. The remaining limits, $L^-_2, L^-_3, \ldots$ are found recursively as in the single support case. Thus, $L^-_2 = g(L^-_0, L^-_1, L^-_2)$, and so this limit exists and is unique, etc. The verification of a recursion similar to (1) is straightforward in this case.

The finite support right-diagonal case is handled precisely as the right-diagonal case of the single support configuration except that we distinguish the column $\mathcal{V}_q$ and, without loss of generality, consider all right-diagonal sequences as originating on $\mathcal{V}_q$. This only leaves out a finite number of terms and so the asymptotic dynamics are not affected. In this case one can show that $L^+_0 = x^0_q$, L^+_1 exists and is unique, and as before there holds a relation similar to (2) for all subsequent limits each one of which can be calculated explicitly on account of (I). Vertical sequences are handled as before so that, generally, one can only guarantee the existence of various limit points, even for sequences in the "dark area", that is that area enclosed by those columns between $\mathcal{V}_{-k}$ and $\mathcal{V}_q$.

Remark. All but nine fuzzy rules (mentioned above) satisfy the conditions of continuity and (I), (II) above, so the analysis captures much of the dynamics of essentially every single rule. The exceptions thus noted are distinguished by the fact that their diagonal function has every point in $[0,1]$ as a fixed point! More refined estimates as to the rate of convergence of a diagonal, questions of convergence in the dark area, etc. may be obtained on a case-by-case basis. It follows that there are no random or chaotic fuzzy rules in this context (except for the 9 undetermined ones) since all existing limits are continuous functions of the initial data. Chaos can occur when the iterates $G^m(a)$ fail to converge

thus violating (II) or, if they do converge, they admit sensitive dependence upon a because the original *nonlinear* rule (not necessarily related to a fuzzy CA) admits chaotic sequences under iterations. For example, the "rule" defined by $g(x, y, z) = 4x(1-x^2)$ produces a chaotic sequence in the case of a single support initial configuration with fixed cell value $a \in (0, 1)$. The spatio-temporal evolution of fuzzy rule 30 can be found in [3] and these results follow immediately from our methods. We note that the long term dynamics of fuzzy rule 110 were obtained in [10] using special arguments pertaining to the form of the rule itself along with its representation as a Taylor polynomial. This also follows from our methods.

2.3 The Dynamics of Fuzzy Rule 18

As for new phenomena we exhibit, for example, the long term dynamics of fuzzy rules 18, 45 and 184 (a typical representative of the exceptional list of fuzzy rules that defy (I)) below, in this and the next subsections. The canonical expression for fuzzy rule 18 is given by $g_{18}(x, y, z) = (1 - y)(x + z - 2xz)$. The methods presented above can be applied easily here so the following result is presented without proof.

Theorem 1 *Let $a \in (0, 1)$ be a single seed in a zero background. Then the long term dynamics of fuzzy rule 18 are given as follows:*

- $L_0^-(a) = a$, $L_1^-(a) = 0$, *and* $L_{2n}^-(a) = 1/2$, *for each* $n \geq 1$, *while* $L_{2n+1}^-(a) = 0$, *for each* $n \geq 0$.
- $L_0^+(a) = a$, $L_1^+(a) = 0$, *and* $L_{2n}^+(a) = 1/2$, *for each* $n \geq 1$, *while* $L_{2n+1}^+(a) = 0$, *for each* $n \geq 0$.
- *Vertical columns fail to converge as they always have two limit points, either 0 or* $1/2$.

Some may argue that the vertical columns actually represent asymptotically periodic sequences. Either way, there is no definite convergence. The symmetry about the central column in the evolution of rule 18 is due to the relation $g(x, y, z) = g(z, y, x)$ satisfied by this rule for every $x, y, z \in \mathcal{U}$. The asymptotics of the finite support case are governed by the first and last cells of the initial configuration and rule 18's dynamics are identical, by the theory above, to those of the single cell case.

2.4 The Dynamics of Fuzzy Rule 45

In order not to focus our examples on *even* rules we present an example of the application of our techniques to the evolution of fuzzy rule 45, an *odd* rule. The canonical form of this rule is given by the expression $g_{45}(x, y, z) = 1 - x - z + yz + 2xz - 2xyz$. Its *diagonal function* $d(x) \equiv g_{45}(x, x, x)$ is given by $d(x) = -2x^3 + 3x^2 - 2x + 1$. It has only one real fixed point, $x = 1/2$, which is attracting. For a single seed, a, in a zero background observe that, by induction, $x_{-2n}^{2n} = 0 = \ldots = x_{n+1}^{2n}$ for each $n \geq 1$. In addition, $x_{-(2n+1)}^{2n+1} = 1 = \ldots =$

$x^{2n+1}_{-(n+2)}$ for $n \geq 1$. Thus, no left-diagonal sequence converges in the strict sense although it is a simple matter to see that we have eventually periodic behavior (or an eventual 2-cycle) for the left diagonals. The right-diagonals lead to interesting phenomena. Note that condition (I) is untenable for this rule, that is, the limit L_0^+ does not exist (since the zeroth right-diagonal sequence alternates between 0 and 1, or represents a 2-cycle), and the same can be said of L_1^+ (since the first right-diagonal sequence alternates between a and $1-a$, another 2-cycle). However, the limit L_2^+ *does* exist and, in fact, $L_2^+ = 1/2$. The following proof of the preceding result is typical for odd rules. Let n be an even integer. Then

$$x^n_{n-2} = g_{45}(x^{n-1}_{n-3}, x^{n-1}_{n-2}, x^{n-1}_{n-1}). \tag{3}$$

However, for even n, $x^{n-1}_{n-1} \to 1$ while $x^{n-1}_{n-2} \to 1-a$. Taking the limit in (3) we get that $L_2^{even} = g_{45}(L_2^{even}, 1-a, 1)$. Solving for L_2^{even} we get that $L_2^{even} = 1/2$, as stated. If n is an odd integer, then (3) is still in force, but $x^{n-1}_{n-2} \to a$ while $x^{n-1}_{n-1} \to 0$ as $n \to \infty$ through odd numbers. It follows that $L_2^{odd} = g_{45}(L_2^{odd}, a, 0)$. Solving for L_2^{odd} we find that $L_2^{odd} = 1/2$ as well. From this we see that the common limit $L_2^{even} = L_2^{odd}$ is, in fact the limit, $L_2^+ = 1/2$. To find L_3^+ we proceed as usual, noting that the non-existence of L_1^+ is unimportant. For example, if $a < 1$, passing to the limit inferior in (3) we deduce that $L^+_{3,inf} = g_{45}(L^+_{3,inf}, L_2^+, a)$ where $L^+_{3,inf}$ is the limit inferior of the sequence x^n_{n-2}. Observe that one can solve for $L^+_{3,inf}$ uniquely in the preceding display provided $a \neq 2$, which is necessarily the case (since the range of g_{45} is contained in $[0,1]$). The unique value thus obtained is $L^+_{3,inf} = 1/2$. A similar argument using the limit superior gives us that $L^+_{3,sup} = 1/2$. Since these two limits agree, the sequence itself has a limit and this limit must be $L_3^+ = 1/2$. We see that $L_k^+ = 1/2$, for each $k \geq 3$, by induction.

2.5 The Dynamics of Fuzzy Rule 184

We consider the dynamics of a rule that fails to satisfy (I). As we pointed out earlier there are exactly nine such (so-called exceptional) rules, including the present one. The canonical form of rule 184 is given by the expression $g_{184}(x,y,z) = x - xy + yz$. Its *diagonal function* $d(x) \equiv g_{45}(x,x,x)$ is given by $d(x) = x$. Thus, *every* real number in $[0,1]$ is a fixed point (it is *this* fact that characterizes the other exceptional rules). Next, for a single seed a in a zero background observe that, by induction, $x^n_n = a$ for each $n \geq 1$, so that this rule is a right-shift. Clearly, for a single seed its dynamics are trivial. The difficulty occurs when we pass to the case of finite support/random initial data.

Consider the case of two seeds, $a, b \in (0,1)$ in a zero background. We take it that $x^0_0 = a, x^0_1 = b$. Of special interest in this case is the convergence of the right-diagonals and the dynamics of the rule along them. Note that $x^{n-1}_n = b$ for all $n \geq 1$, so that the limit of this sequence (or zeroth diagonal, $\mathcal{V}_0$), is $L_0^+ = b$. Next, the terms of the first right-diagonal, $\mathcal{V}_1$, are given by $x^n_n = a(1-b)^{n-1}$, a result that is easily verified by induction. It follows that its limit, $L_1^+ = 0$, except in the special case where $b = 0$, in which case this reduces to the single

seed scenario already discussed above. Difficulties arise in the discussion bearing on the next diagonal, $\mathcal{V}_2$. Applying our technique to this situation we find $L_2^+ = g_{184}(L_2^+, L_1^+, L_0^+) = g_{184}(L_2^+, 0, b) = L_2^+$. Thus, no *a priori* information regarding L_2^+ is obtainable using our methods, as they stand.

In order to circumvent this difficulty we suggest the following approach. We suspect that L_2^+ may be obtained by passage to the limit of a small parameter $\varepsilon > 0$ using the claim that $L_2^+ = g_{184}(L_2^+, \varepsilon, b)$, holds for every $\varepsilon > 0$. This then results in the equality $L_2^+ \varepsilon = b\varepsilon$, from which we conclude that $L_2^+ = b$. This argument is supported by numerical calculations of this specific limit. Since $L_2^+ = b$ we get $L_3^+ = g_{184}(L_3^+, L_2^+, L_1^+) = g_{184}(L_3^+, b, 0) = L_3^+ - bL_3^+$ and so $L_3^+ = 0$. Continuing in this way we find the sequence of limits, $L_k^+ = b$ if k is even, and $L_k^+ = 0$ if k is odd. Once again these limiting values are supported by calculations in this two-seed case.

Remark. We note that rigorous justification for our technique of *passage to the limit of a small parameter* is lacking, except for the fact that it yields the correct dynamics for this rule. We suspect that this yields correct limiting behavior for finite random initial data in rule 184. However, it is an open question whether this technique will produce the correct limiting behavior for the other eight remaining exceptional fuzzy rules.

Concluding Remarks. An iterative method for finding the dynamics of all fuzzy CA as defined in [4] is produced. This gives a road-map for determining the global evolution of a general fuzzy rule. It is likely that all fuzzy CA lead to deterministic behavior and there are no chaotic or random rules (except possibly for rules 170, 172, 184, 202, 204, 216, 226, 228, and 240; see [7]). The dynamics of these nine rules remain undetermined at this time even though the methods used in rule 184 above may be used, it is not clear that this will work in all cases. Minor modifications show that the techniques presented here apply to neighborhood structures with an arbitrary fixed number of cells (*e.g.,* 5 or 7) and finite support (or random) initial configurations.

Acknowledgments. I am grateful to Richard Phillips [11] of Wolfram Corp. for producing very useful output for random initial strings of all fuzzy rules.

References

1. Andrew Wuensche, Personal communication, *Complex Systems 98 Conference*, University of New South Wales, Australia, Fall, 1998.
2. P. Flocchini, F. Geurts, A. Mingarelli, N. Santoro, *Convergence and aperiodicity in fuzzy cellular automata: revisiting rule 90*, Physica D, **142** (2000), 20–28.
3. Angelo B. Mingarelli and Elzbieta Beres, *The dynamics of fuzzy cellular automata: Rule 30*, WSEAS Trans. Circuits and Systems (10) **3** (2004), 2211-2216.
4. G. Cattaneo, P. Flocchini, G. Mauri, C. Quaranta Vogliotti, N. Santoro, *Cellular automata in fuzzy backgrounds*, Physica D **105** (1997), 105-120
5. C. A. Reiter, *Fuzzy automata and life*, Complexity, **7** (3) (2002), 19-29.

6. P. Maji, R. Nandi, P. P. Chaudhuri, *Design of fuzzy cellular automata (FCA) based pattern classifier*, in *Fifth International Conference on Advances in Pattern Recognition*, ICAPR-2003, December 10-13, 2003, Calcutta, India. To appear.
7. G. Cattaneo, P. Flocchini, G. Mauri, and N. Santoro, *Fuzzy cellular automata and their chaotic behavior*, in Proc. International Symposium on Nonlinear Theory and its Applications, Hawaii, IEICE Volume 4, (1993) 1285-1289
8. S. Wolfram, *A New Kind of Science*, Wolfram Media, Champaign, Il., 2002.
9. F. John, *Partial Differential Equations*, Third Edition, Springer-Verlag, New York, 1980, ix, 198 p.
10. Angelo B. Mingarelli, *Fuzzy rule 110 dynamics and the golden number*, WSEAS Trans. Computers, 2 (4) (2003), 1102-1107.
11. Richard Phillips, Steve Wolfram Science Group, Wolfram Corp., *personal communications*, 2004.

A Cellular Automaton SIS Epidemiological Model with Spatially Clustered Recoveries

David Hiebeler

Dept. of Mathematics and Statistics,
333 Neville Hall, University of Maine,
Orono, ME 04469-5752 USA
hiebeler@math.umaine.edu

Abstract. A stochastic two-state epidemiological cellular automaton model is studied, where sites move between susceptible and infected states. Each time step has two phases: an infectious phase, followed by a treatment or recovery phase. During the infectious phase, each infected site stochastically infects its susceptible neighbors. During the recovery phase, contiguous blocks of sites are reset to the susceptible state, representing spatially clustered treatment or recovery. The spatially extended recovery events are coordinated events over groups of cells larger than standard local neighborhoods typically used in cellular automata models. This model, which exhibits complex spatial dynamics, is investigated using simulations, mean field approximations, and local structure theory, also known as pair approximation in the ecological literature. The spatial scale and geometry of recovery events affects the equilibrium distribution of the model, even when the probability of block recovery events is rescaled to maintain a constant per-site recovery probability per time step. Spatially clustered treatments reduce the equilibrium proportion of infected invididuals, compared to spatially more evenly distributed treatment efforts.

1 Introduction

Consider a discrete-time lattice-based epidemiological model, where each site can be in one of two states: susceptible and infected. Infection and recovery parameters are ϕ and μ, respectively. During each time step, the following two things occur, in this order:

- Infection: every infected site will infect each of its susceptible neighbors, independently with probability ϕ each. The standard von Neumann neighborhood was used, consisting of the four orthogonal neighbors of a site. If an infected site tries to infect an already-infected neighbor, there is no effect.
- Recovery: contiguous blocks of sites recover simultaneously. Parameters b_1 and b_2 specify the dimensions of recovery blocks. Each block will consist of a $b_1 \times b_2$ (rows $\times$ columns) block of sites or a $b_2 \times b_1$ block of sites, each with

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 360–367, 2005.
© Springer-Verlag Berlin Heidelberg 2005

probability 0.5. During the recovery phase, each site, independently with probability γ (computed from μ as described in the section "Pair Approximations" below), will have a recovery block placed so that its upper-left corner is located at the target site being considered. Note that multiple recovery blocks within a time step may spatially overlap.

This is a discrete-time analogue of a continuous-time population model investigated in [1]. While qualitative results are similar in the two models, many of the details of the analysis differ, being more complex for the discrete-time model.

2 Simulations

Simulations were performed on a 300×300 lattice with wraparound (toroidal) boundary conditions. After each time step, the proportions of sites in the susceptible and infected states were recorded. Based on methods used by [2] to test for equilibrium, beginning on time step 1000, on every time step a least-squares regression line was fit to the most recent 100 measurements of the proportion of sites infected. When the slope of this line was less than 0.001, and the difference between the minimum and maximum proportion of infected sites over the previous 100 time steps was less than 0.03, the system was considered to have reached equilibrium. After it was determined that equilibrium was reached, the simulation was run for another 50 time-steps, and the proportion of infected sites was averaged over those final 50 time steps and recorded as the final proportion of infected sites for the simulation. Although exploration showed that the model was not sensitive to initial conditions, in order to reduce the time needed to reach equilibrium, the equilibrium predicted by the local-dispersal mean-field approximation [3] was used as the initial proportion of infected individuals. However, if this initial proportion was less than 0.1, then 0.1 was used instead, to prevent fixation to a lattice completely in the susceptible state solely due to fluctuations from an initial small population of infected sites.

3 Pair Approximation

Let 0 represent the susceptible state, and 1 the infected state. The state of the lattice can be approximately described by the probabilities $P[ij]$ (where $i, j \in \{0, 1\}$) that a pair of adjacent sites are in the state configurations 00, 01, 10, and 11. Assuming rotational symmetry, $P[01] = P[10]$ (as well as $P\begin{bmatrix} i \\ j \end{bmatrix} = P[ij]$), and using the fact that the four probabilities must sum to one, only two independent probabilities are needed to describe the state of the system, for example $P[00]$ and $P[01]$. The other two probabilities may then be computed as $P[10] = P[01]$ and $P[11] = 1 - P[00] - P[01] - P[10] = 1 - P[00] - 2P[01]$. Marginal probabilities of the possible states for a single site can be recovered by summing over block probabilities, $P[i] = P[i0] + P[i1] \quad \text{for } i \in \{0, 1\}$.

As described in [3], based on ideas explored in [4, 5], the block probabilities $P_{t+1}[ij]$ at time $t+1$ can be estimated using the current probabilities $P_t[ij]$ by first estimating the probabilities of all pre-images of a pair of sites, and then applying the cellular automaton rule to those pre-images and using the law of total probability. A pre-image of a pair of sites is a set of state configurations of the group of sites which the target pair of sites depends on when updating its states, as shown in Fig. 1.

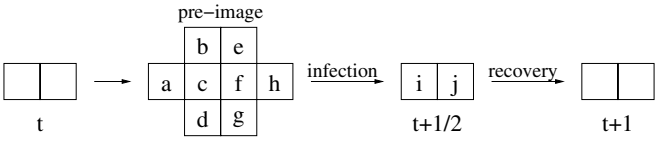


Fig. 1. The group of sites in a pre-image of a pair of sites is shown. A pre-image is the set of states of all sites which the target pair of sites depend on when updating their states, i.e. all neighbors of the pair of target sites. The probabilities of all pre-images are estimated, and then used to compute probabilities of all state configurations of a pair of sites after the infectious phase of a time step, $P_{t+\frac{1}{2}}[ij]$. The probabilities after the recovery phase, $P_{t+1}[ij]$ are then computed

The pair approximation used here assumes that non-adjacent sites are independent when conditioned on any shared neighbors, i.e.

$$P\begin{bmatrix} j & k \\ i & \cdot \end{bmatrix} = P[ijk] = P[ij\cdot]P[\cdot\,k|ij\cdot] = P[ij]P[\cdot\,k|\,\cdot\,j\cdot] = \frac{P[ij]P[jk]}{P[j]}. \quad (1)$$

In the expression above, the conditional probability that the third site is in state k given the states of the other two sites are i and j does not depend on the first site's state because the first and third sites are not adjacent. See e.g. [6, 7] for a discussion of these methods applied to continuous-time epidemiological models. Note that hereafter, $0/0$ is defined to be 0 in (1) when extending block probabilities, since if $P[j] = 0$, then $P[ij] = 0$ and $P[ijk] = 0$ for all $i, k \in \{0, 1\}$.

Heuristically, the probability of a 3×1 block may be built up by covering it with two overlapping 2×1 blocks, multiplying the 2×1 block probabilities and dividing by the overlapping single-site probability. The 2×1 probabilities may be repeatedly extended in this manner to build up probabilities of ever-larger blocks [3, 8]. However, as also seen with many information-theoretic measures of spatial complexity [9], in two or more dimensions, there can be more than one way to cover larger blocks of sites with 2×1 sites [3], and thus there is not a unique way to estimate the probabilities of pre-images. This can be seen when trying to compute the probability of a 2×2 block:

$$\begin{aligned} P\begin{bmatrix} a & b \\ c & d \end{bmatrix} &= P\begin{bmatrix} a & b \\ \cdot & d \end{bmatrix} P\left[\begin{array}{cc|cc} \cdot & \cdot & a & b \\ c & \cdot & \cdot & d \end{array}\right] = \frac{P[ab]P[bd]}{P[b]} P\left[\begin{array}{cc|cc} \cdot & \cdot & a & \cdot \\ c & \cdot & \cdot & d \end{array}\right] \\ &= \frac{P[ab]P[bd]}{P[b]} \frac{P\begin{bmatrix} a & \cdot \\ c & d \end{bmatrix}}{P\begin{bmatrix} a & \cdot \\ \cdot & d \end{bmatrix}} = \frac{P[ab]P[bd]}{P[b]} \frac{P\begin{bmatrix} a & \cdot \\ c & d \end{bmatrix}}{\sum_{i\in\{0,1\}} P\begin{bmatrix} a & \cdot \\ i & d \end{bmatrix}} \end{aligned} \quad (2)$$

where the first probability has been expanded using (1), and the second probability has been approximated by assuming that the site labelled c does not depend on the non-adjacent site labelled b, and then expanding the results using the elementary definition of conditional probability. The sum in the denominator of the final expression may be calculated using (1). The non-uniqueness of this method may be seen by observing that in the calculation above, c was the "last" site added into the block when expanding the 2×2 block probability using a conditional probability; if instead d were the last site considered, a different expression would result. There is no clear way to choose one method over the other; one could choose whichever term maximizes entropy of the resulting block probabilities, but (2) was used in this study.

Because of the nature of the update rule used, computation of the new probabilities $P_{t+1}[ij]$ in terms of the current probabilities $P_t[ij]$ may also be broken into the two phases of infection and recovery.

For the infectious phase, the probabilities of all possible pre-images as shown in the center of Fig. 1 are needed. Following the discussion above, the probability extension used here was

$$P\begin{bmatrix} \cdot & b & e & \cdot \\ a & c & f & h \\ \cdot & d & g & \cdot \end{bmatrix} = \frac{P[cf]P[bc]P[fg]P[ac]P[fh]P[be]P[ef]P[cd]P[dg]}{(P[c])^2(P[f])^2P[e]P\begin{bmatrix} b & \cdot \\ \cdot & f \end{bmatrix}P[d]P\begin{bmatrix} c & \cdot \\ \cdot & g \end{bmatrix}} \tag{3}$$

where

$$P\begin{bmatrix} b & \cdot \\ \cdot & f \end{bmatrix} = \sum_{i\in\{0,1\}} P\begin{bmatrix} b & \cdot \\ i & f \end{bmatrix} = \sum_{i\in\{0,1\}} P[bif] = \sum_{i\in\{0,1\}} \frac{P[bi]P[if]}{P[i]}$$

and similarly for $P\begin{bmatrix} c & \cdot \\ \cdot & g \end{bmatrix}$. The probabilities of pre-images given by (3) may therefore be estimated using only the current 2×1 block probabilities.

Once the pre-image probabilities have been estimated, the probabilities $P_{t+\frac{1}{2}}[ij]$ after the infectious phase of the time step may then be computed, by conditioning on the pre-image at time t:

$$P_{t+\frac{1}{2}}[ij] = \sum_{G\in\mathcal{G}} P_t(G)P(G \to [ij]) \tag{4}$$

where $\mathcal{G}$ is the set of all pre-images, $P_t(G)$ is the probability of pre-image G, and $P(G \to [ij])$ is the probability that pre-image G results in the state $[ij]$ for the target pair of sites after the infectious phase. Because there are 8 sites in the pre-image, and two states per site, there are $2^8 = 256$ pre-images in total. Because only infections occur during this phase, if $c = 1$ and $i = 0$, or $f = 1$ and $j = 0$ in Fig. 1, then $P(G \to [ij]) = 0$. Otherwise, the probability will be based on binomial distributions. Let $k_L(G) = a+b+d+f$ be the number of neighbors of the left site c which are occupied in the pre-image G, and $k_R(G) = c+e+g+h$ be the number of neighbors of the right site f which are occupied.

- If $c = 0$ and $f = 0$, then $P(G \to [00]) = (1-\phi)^{k_L(G)}(1-\phi)^{k_R(G)}$, and $P(G \to [01]) = (1-\phi)^{k_L(G)}(1-(1-\phi)^{k_R(G)})$.
- If $c = 0$ and $f = 1$, then $P(G \to [00]) = 0$, and $P(G \to [01]) = (1-\phi)^{k_L(G)}$.
- If $c = 1$ and $f = 0$, or if $c = 1$ and $f = 1$, then $P(G \to [00]) = P(G \to [01]) = 0$.

Once the probabilities $P_{t+\frac{1}{2}}[ij]$ have been estimated using (4) together with the above information, the final probabilities $P_{t+1}[ij]$ may then be estimated by applying the recovery phase of the cellular automaton rule. Because the application of recovery blocks is externally imposed and does not depend on the current states of cells or their neighbors, extension of block probabilities is not needed for this phase. Only the application of basic probability is needed, to compute the probabilities that among a pair of sites, neither, one, or both sites are contained within a recovery block. Sites may be part of a recovery block if any of several nearby sites are the target of such a block. For example, consider the case where $b_1 = 2$ and $b_2 = 3$, i.e. 2×3 and 3×2 recovery blocks are used. For the pair of sites drawn in bold in Fig. 2, recovery blocks at any of the labelled sites will affect one or both sites in the pair, as follows:

- Both 2×3 and 3×2 blocks targetted at sites A will affect only the right site of the pair.
- Both 2×3 and 3×2 blocks targetted at sites B will affect both sites of the pair.
- 3×2 blocks at site C will affect the right site of the pair, but 2×3 blocks will not affect the pair.
- 3×2 blocks at site D will affect both sites of the pair, but 2×3 blocks will not affect the pair.
- 2×3 blocks at sites E will affect both sites of the pair, but 3×2 blocks will affect only the left site of the pair.
- 2×3 blocks at sites F will affect the left site of the pair, but 3×2 blocks will not affect the pair.
- 3×2 blocks at site G will affect the left site of the pair, but 2×3 blocks will not affect the pair.

Similar enumerations can be performed for any values of b_1 and b_2. This information may then be used to calculate the probabilities that particular sites in a pair are affected by one or more blocks. Such calculations show that the probability that both sites in a pair will be affected by one or more recovery blocks is

$$P([11] \to [00]) = 1 + A^{c_1} B^{c_2} \left(A^{c_3} B^{c_4} - 2\right) \tag{5}$$

where $A = 1-\gamma$, $B = 1-\gamma/2$, $c_1 = (b_{\min})^2$, $c_2 = 2(b_{\max} - b_{\min})b_{\min}$, $c_3 = b_{\min}$, and $c_4 = b_{\max} - b_{\min}$, $b_{\min} = \min(b_1, b_2)$ and $b_{\max} = \max(b_1, b_2)$. Similarly, the probability that the left site in a pair will be affected by one or more recovery blocks, but that the right site will not be affected by any blocks, is

$$P([11] \to [01]) = A^{c_1} B^{c_2} \left(1 - A^{c_3} B^{c_4}\right) \tag{6}$$

Combining the two, the probability that any single site will be affected by one or more recovery blocks is

$$P([1] \to [0]) = 1 - A^{c_1} B^{c_2} \tag{7}$$

We wish the single-site recovery probability to be μ, but because a site may recover due to being hit by recovery blocks targeted at any number of neighboring sites, the recovery probability is altered. To correct for this, we use (7) to define $f(\gamma) = 1 - (1-\gamma)^{c_1}(1-\gamma/2)^{c_2}$, and then numerically solve for the value γ satisfying $f(\gamma) = \mu$. This value γ is the adjusted recovery probability used in all simulations and approximations of the model; it is the recovery block probability which yields a single-site recovery probability of μ per time step.

	G	D	C
F	E	B	A
F	E	**B**	**A**

Fig. 2. The set of all sites where a 2×3 recovery block could be targeted and affect a specific pair of sites (shown in bold at lower-right corner). See text for explanation of site labels

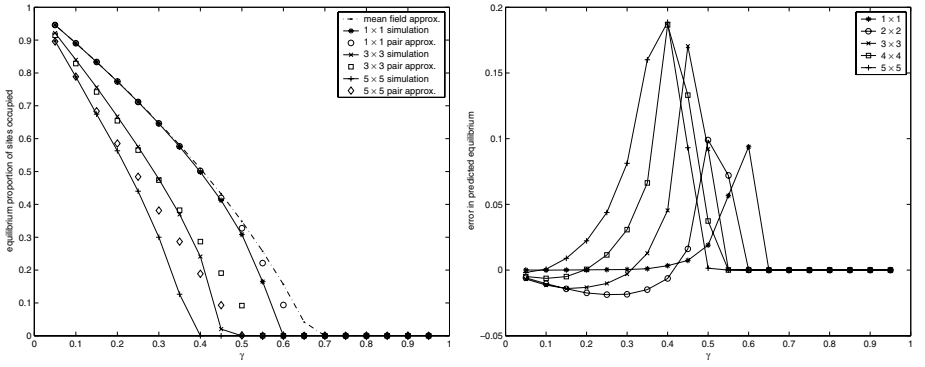
This adjustment of recovery rates is simpler in continuous-time models, where the recovery rate merely needs to be divided by $b_1 b_2$, the size of the recovery blocks [1]. In the discrete-time model, however, if this rescaling is used, the number of recovery blocks affecting a single site approaches a Poisson distribution as the block sizes become large, and the single-site recovery probability approaches $1 - e^{-\mu}$, thus making the more complex rescaling above necessary.

The recovery probabilities above in (5)–(7) may be combined with the infection probabilities given by (4) to compute the updated probabilities on the next time step, as follows:

$$P_{t+1}[00] = P_{t+\frac{1}{2}}[00] + 2P_{t+\frac{1}{2}}[01]P([1] \to [0]) + P_{t+\frac{1}{2}}[11]P([11] \to [00])$$
$$P_{t+1}[01] = P_{t+\frac{1}{2}}[01](1 - P([1] \to [0])) + P_{t+\frac{1}{2}}[11]P([11] \to [01])$$

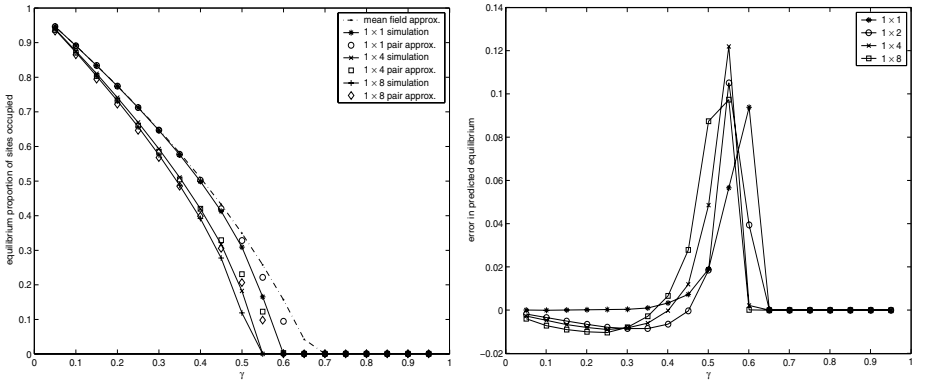
4 Results

Equilibrium proportions of sites infected are shown for $\phi = 0.5$, as the per-site recovery rate μ was varied between 0 and 1, for $n \times n$ recovery blocks in Fig. 3 and $1 \times n$ recovery blocks in Fig. 4. Results are shown for simulations, pair approximations, and the mean-field approximation, for which the rate of recovery only depends on the single-site rate given by (7). Note that the mean-field approximation is not dependent on the size of recovery blocks because it is a spatially implicit method which ignores all spatial correlations; thus only one mean-field curve appears in the figures. Errors in the predictions, i.e. the pair



(a) (b)

Fig. 3. Results using square $n \times n$ recovery blocks of various sizes, as the per-site recovery rate μ varies between 0 and 1 on the x-axis, with $\phi = 0.5$. (a) The equilibrium proportion of infected sites is shown, from simulations, pair approximations, and the mean field approximation. (b) Prediction error, i.e. predictions from pair approximations minus measurements from simulations



(a) (b)

Fig. 4. Results using long $1 \times n$ recovery blocks of various sizes. Compare with Fig. 3. (a) Equilibrium proportion of infected sites. (b) Pair approximation prediction error

approximation minus the simulation measurements, are also shown in Figs. 3 and 4.

It can be seen from the figures that the geometry of recovery events does affect the equilibrium distribution, even when the single-site recovery probability is held constant. This effect is more pronounced for square recovery blocks than for long, narrow blocks. For a disease with only local infection and a regular treatment regime, the long-term prevalence of the disease would be reduced by focusing treatment in fewer contiguous areas, rather than distributing treatment more evenly throughout the population.

As with the continuous-time version of the model, the pair approximations do fairly well at predicting simulation results. They do most poorly near the critical value of the recovery rate at which the equilibrium proportion of infected individuals transitions between 0 and a positive value, when spatial correlations decay more slowly with distance over spatial scales beyond that reflected by the pair approximation [1]. Also, as can be seen in Fig. 3b, the pair approximations become less accurate as the spatial scale of the recovery blocks becomes larger. The pair approximations are more accurate for long $1 \times n$ blocks as compared with square $n \times n$ blocks (compare the scales of the y-axes in Figs. 3b and 4b), and are also more accurate over a wider range of values of the rescaled recovery rate γ. Although in continuous time, pair approximations tend to almost always overestimate the equilibrium proportion of infected sites, in the discrete-time model it can be clearly seen from the figures that the pair approximation underestimates this value over a significant range of the parameter space. Further investigation is needed to determine exactly why the more complex interactions in the discrete-time model give rise to this behavior.

References

1. Hiebeler, D.: Spatially correlated disturbances in a locally dispersing population model. Journal of Theoretical Biology **232** (2005) 143–149
2. Caswell, H., Etter, R.J.: Ecological interactions in patchy environments: From patch occupancy models to cellular automata. In Levin, S., Powell, T., Steele, J., eds.: Patch Dynamics, Springer-Verlag (1993) 93–109
3. Hiebeler, D.: Stochastic spatial models: From simulations to mean field and local structure approximations. Journal of Theoretical Biology **187** (1997) 307–319
4. Gutowitz, H.A., Victor, J.D.: Local structure theory in more than one dimension. Complex Systems **1** (1987) 57–68
5. Wilbur, W.J., Lipman, D.J., Shamma, S.A.: On the prediction of local patterns in cellular automata. Physica D **19** (1986) 397–410
6. Levin, S.A., Durrett, R.: From individuals to epidemics. Philosophical Transactions: Biological Sciences **351** (1996) 1615–1621
7. Filipe, J., Gibson, G.: Comparing approximations to spatio-temporal models for epidemics with local spread. Bulletin of Mathematical Biology **63** (2001) 603–624
8. Gutowitz, H.A., Victor, J.D., Knight, B.W.: Local structure theory for cellular automata. Physica D **28** (1987) 18–48
9. Feldman, D.P., Crutchfield, J.P.: Structural information in two-dimensional patterns: Entropy convergence and excess entropy. Physical Review E **67** (2003)

Simulating Market Dynamics with CD++

Qi Liu and Gabriel Wainer

Department of Systems and Computer Engineering, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6 Canada
{liuqi, gwainer}@sce.carleton.ca

Abstract. CD++ is an implementation of the Cell-DEVS formalism, which has been used to simulate various complex systems. In this study, we constructed a Cell-DEVS to simulate the dynamics of a dual market. Using new features of CD++, we obtained accurate results taking into account consumers' purchasing history. The resulting model allows fast execution, easier model implementation and maintenance.

1 Introduction

Cellular Automata (CA) [1] have become popular to simulate complex systems in a variety of research areas. CA are infinite n-dimensional lattices of cells updated synchronously according to local rules. Cell-DEVS [2], instead, uses the DEVS (Discrete Events Systems specifications) formalism [3] to define a cell space where each cell is defined as an atomic DEVS model. Each cell receives input external events from its neighboring cells and executes the events by evaluating local computing rules. The cell will change its states according to the execution results after a delay, and when it changes, it sends output messages to all its neighbors. CD++ [4] is an implementation of Cell-DEVS, which was recently extended to improve model definition, permitting to define more compact and flexible models [5]. We present a Cell-DEVS model to study the dynamics of markets [6]. The model simulates a dual market where consumers choose among competing products based on their preferences and the influence of others. A cell stands for a consumer who periodically renews the license with one of two Operating System providers. Three factors influence their behavior:

1. $U(c_{ij}, n, t)$, the **utility** consumer c_{ij} can obtain by using OS_n in time t,
2. $E(c_{ij}, n, t)$, the **network externality**, i.e., the influence of others; and
3. $P(c_{ij}, t)$, the **price** that consumer c_{ij} must pay.

The cell space is defined by a coupled Cell-DEVS, whose atomic cells are defined as C_{ij} =<X,Y,I,S,N,d,τ>.X={0,1,2} are the external inputs; Y={0,1,2} are the external outputs; S={0,1,2} are the states (0: non-user; i=1,2: user of OS_i). N={0,1,2} is the set of the inputs; d=100 is the transport delay for each cell. τ:N→S is the local computing function defssssined by Table 1 with $V(c_{ij},1,t)=U(c_{ij},n,t)+E(c_{ij},n,t)-P(c_{ij},t)$, for n=1 or 2 (*U, E* and *P* are computed as in [7], using the rules defined in [6]).

The model was implemented in CD++, and a set of experiments was carried out with six different settings. The tests were categorized into two groups: mature and

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 368 – 372, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Table 1. Local computing rules

Result: 1	Rule : $V(c_{ij}, 1, t) > V(c_{ij}, 2, t)$ AND $V(c_{ij}, 1, t) > 0$
Result: 2	Rule : $V(c_{ij}, 2, t) > V(c_{ij}, 1, t)$ AND $V(c_{ij}, 2, t) > 0$
Result: 0	Rule : $0 > V(c_{ij}, 1, t)$ AND $0 > V(c_{ij}, 2, t)$

new markets. The former group uses an initial cell space where the three possible states for each cell are uniformly distributed to represent a mature market; the latter-group simulates a new market, in which only a few of cells represent new users. Fig. 1 shows the local computing rules for the new market and fluctuating price case.

```
%vo1 and vo2: skill accumulation for OS1 and OS2 respectively
%current state=0, vo1 & vo2 are depreciated by 0.25 at each time step
rule:{if((stateCount(1)+9*$vo1)>(stateCount(2)+9*$vo2),1,
       if((stateCount(2)+9*$vo2)>(stateCount(1)+9*$vo1), 2, 0))}
       {$vo1:=$vo1*0.25; $vo2:=$vo2*0.25;}  100  {(0,0)=0}
%current state=1: vo1 is incremented before depreciation
rule:{if((stateCount(2)+9*$vo2)>(stateCount(1)+9*$vo1),2,1)}
       {$vo1:=($vo1+1)*0.25; $vo2:=$vo2*0.25;}  100  {(0,0)=1}
%current state=2: vo2 is incremented by 1 before depreciation
rule:{if((stateCount(1)+9*$vo1)>(stateCount(2)+9*$vo2),1,2)}
       {$vo1:=$vo1*0.25; $vo2 := ($vo2+1)*0.25;}  100  {(0,0)=2}
```

Fig. 1. Definition of local computing rules in CD++: mature market and same price scenario

Three pricing strategies are defined: products with the same price, products with different prices, and products with fluctuating prices. The local computing rules are instantiated using the parameter values defined in Table 2.

Table 2. Parameter values for experimental frames

Settings	Mature Market			New Market		
	= Price	≠ Price	Fluctuating Price	= Price	≠ Price	Fluctuating Price
Parameters	U_{max}=.8 U_{min}=.4 θ=λ=.5 P_{OS1}=.25 P_{OS2}=.25	U_{max}=.8 U_{min}=.4 θ=λ=.5 P_{OS1}=.25 P_{OS2}=.3	U_{max}=.8,U_{min}=.4 θ=λ=.5 Q1=Q2=.1 $R1_{max}$=$R2_{max}$=.15 $R1_{min}$=$R2_{min}$=.05 μ1=.2, μ2=1	U_{max}=.8 U_{min}=.4 θ=λ=.5 P_{OS1}=.25 P_{OS2}=.25	U_{max}=.8 U_{min}=.4 θ =λ=.5 P_{OS1}=.25 P_{OS2}=.3	U_{max}=.8,U_{min}=.4 θ=λ=.5 Q1=Q2=0 $R1_{max}$=$R2_{max}$=.3 $R1_{min}$=$R2_{min}$=0 μ1=.2, μ2=1

2 Simulation Results

In this section we present simulation results for the different cases defined on Table 2. White cells represent non-users, light gray represent OS1, and dark gray OS2. The results in Fig. 2 show that non-users begin to use one of the two products with ap-

proximately equal probability, and users using the same products tend to aggregate together to form their own society, which in turn enhances network externality.

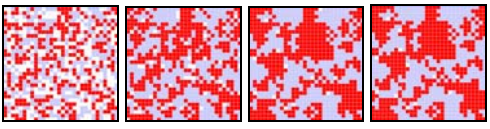


Fig. 2. Mature market and same price scenario

In Fig. 3, the price for OS1 is lower than the price for OS2 (all other parameters fixed), and most of the non-users choose to use OS1. Network externality again results in the aggregation of users.

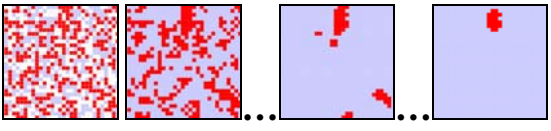


Fig. 3. Mature market and different price scenario

In Fig. 4, OS2 has higher pricing flexibility ($\mu 2 = 1$), while OS1 offers more rigid prices ($\mu 1 = 0.2$). As a result, OS2 gains bigger market share. If the local market shares for both products are equal, the price fluctuation disappears and network externality becomes the sole force in determining consumers' decisions.

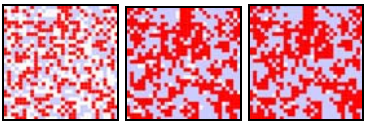


Fig. 4. Mature market and fluctuating price scenario

Fig. 5 shows that the development of the new market starts around the few initial users where the network externality takes effect. The number of users of both products rapidly grows almost at the same rate until the market is saturated. The initial users have been the pivots of a new market.

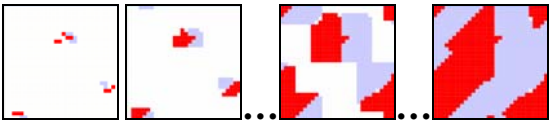


Fig. 5. New market and same price scenario

Fig. 6 shows how OS1 rapidly monopolizes the whole market by virtue of its lower prices (sensitivity of price is high in a new market). Fig. 7 shows that two types of new users ripple out from the initial ones into alternating circles. The development of the market exhibits a pattern that cannot be explained by any single factor of the value function.

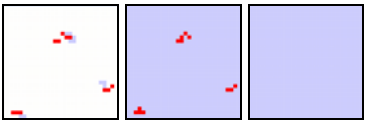


Fig. 6. New market and different price scenario

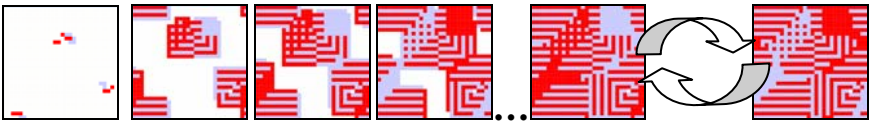


Fig. 7. New market and fluctuating price scenario

3 Conclusion

Cell–DEVS allows describing complex systems using an n-dimensional cell-based formalism. Timing behavior for the cells in the space can be defined using very simple constructions. The CD++ toolkit, based on the formalism, entitles the definition of complex cell-shaped models. We used CD++ to enhance a previous model for simulating market dynamics. The enhanced model obtained more accurate simulation results. Thanks to these new capabilities, we can achieve more efficient simulation, easier model implementation and maintenance. By running the simulation under different settings and analyzing the data generated, statistical results (long-run percentage of market shares, revenue and gross profit, parameters for pricing strategies etc.) can be obtained with sufficient precision. These results can guide us in predicting how the market will respond to various changes.

References

1. Wolfram, S. 2002. A new kind of science. Wolfram Media, Inc
2. G. Wainer, N. Giambiasi: N-dimensional Cell-DEVS. Discrete Events Systems: Theory and Applications, Kluwer, Vol.12. No.1 (January 2002) 135-157
3. B. Zeigler, T. Kim, H. Praehofer: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press (2000).
4. G. Wainer: CD++: a toolkit to define discrete-event models. Software, Practice and Experience. Wiley. Vol. 32. No.3. (November 2002) 1261-1306

5. López, G. Wainer. Improved Cell-DEVS model definition in CD++. P.M.A. Sloot, B. Chopard, and A.G. Hoekstra (Eds.): ACRI 2004, LNCS 3305. Springer-Verlag. 2004.
6. S. Oda, K. Iyori, M. Ken, and K. Ueda, The Application of Cellular Automata to the Consumer's Theory: Simulating a Duopolistic Market. SEAL'98, LNCS 1585. pp. 454-461, Springer-Verlag. 1999.
7. Q. Liu, G. Wainer. Modeling a duopolistic market model using Cell-DEVS. Technical Report SCE-05-04. Systems and Computer Engineering. Carleton University. 2005.

A Model of Virus Spreading Using Cell-DEVS

Hui Shang and Gabriel Wainer

Department of Systems and Computer Engineering, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6 Canada
{shanghui, gwainer}@sce.carleton.ca

Abstract. Cell-DEVS is a combination of CA with the DEVS formalism that allows the definition of cellular models. CD++ is a modeling and simulation tool that implements DEVS and Cell-DEVS. We have used CD++ to build a model about competition between population and viruses. We will discuss how to define such a model in CD++, and will show simulation results under different scenarios.

1 Introduction

In recent years, many simulation models of real systems have been represented using Cellular Automata (CA) [1]. CA are defined as infinite n-dimensional lattices of cells whose values are updated according to local rules. Cell-DEVS [2] was defined as a combination of CA and DEVS (Discrete Events Systems specifications) [3]. A Cell-DEVS model is defined as a lattice of cells holding a state variable and a computing apparatus to update the cell state. Each cell in a Cell-DEVS is a DEVS atomic model, and the cell space is a DEVS coupled model. Once the cell behavior is defined, a coupled Cell-DEVS is created by putting together a number of cells interconnected by a neighborhood relationship. CD++ [4] is a simulation tool based on Cell-DEVS. A built-in specification language provides a set of primitives to define the cell spaces. We have used CD++ to build a Cell-DEVS model on competition between population and viruses, based on the work presented in [5]. The model describes evolution of a population and the interaction between individuals and viruses. Cells valued 1-6 represent individuals (1-young; 2-5: mature; 6: aged). Individuals in cells will use the following rules:

1. *Age increment:* periodically, each cell will be incremented to indicate aging. After reaching the maximum age (6), individual dies (0).
2. *Reproduction:* for each unoccupied cell with at least two adult neighbors (Von Neumann's neighborhood), the cell is set to one.
3. *Movement:* mature individuals can move at random.

Viruses are represented by cells valued 8 (active) or 9 (inactive). They comply with the following rules:

1. *Virus reproduction:* when an unoccupied cell is surrounded by at least one active virus, after a delay, an active virus will occupy the cell.
2. *Virus state change:* after a delay, active virus will become inactive (from 8 to 9). After another delay, the inactive virus will die (from 9 to 0).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 373 – 377, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Individuals and viruses compete for the living space as follows:

1. *Virus killing individuals: if* an individual is surrounded by at least two viruses with vertical or horizontal distribution, the individual and the viruses will die.
2. *Individuals killing virus:* if an active virus is surrounded by at least two individuals, and the viruses have no capacity to kill individuals, the virus dies.
3. *Conflict resolution:* the following conflicts are handled:

- **Empty cell occupation:** an empty cell may be occupied by moving individuals, newborn individuals or active viruses. We give the highest priority to movement, then to reproduction, and lowest priority to virus reproduction.
- **Individual/virus interaction:** if individuals move, they might become out of the range of the virus. In this case, the movement has higher priority.
- **Individual reproduction/movement:** reproduction and movement can happen simultaneously. In that case, the empty cell becomes a newborn individual, while its parents move to other places.

2 Model Execution

The model was implemented in CD++, and a set of experiments was carried out with six different settings. A detailed model definition can be found in [6], and an excerpt can be seen in the Appendix. In the model definition, each cell is associated with cell I/O ports. One of them (*pa*) represents the age of individuals and virus. The other (*pd*) represents the direction of moving individuals. Execution results are shown for different scenarios (gray cells change from light to dark to indicate age; darker cells present active and inactive viruses).

Scenario 1. Population is partially congregated, and viruses are scattered inside the population. No movement rules are applied.

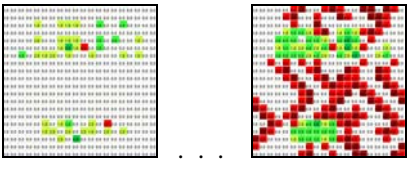


Fig. 1. Virus spread scenario

When compared with the size of population, the number of viruses increases, reflecting the reproduction rules (individuals have more strict rules than viruses). The distribution of population allows individuals to reproduce; nevertheless, the population cannot develop properly due to the action of viruses. The partial congregation of population also provides some space for virus reproduction, so viruses can reproduce quickly. Since the conflict rules give higher priority to population over viruses, it there is a tendency for individuals to congregate.

Scenario 2. Population is packed; viruses are scattered. No movement rules.

The total congregation can prevent the individuals from being killed by viruses, however, it also restrict reproduction. Since viruses scatter inside the population, their reproduction is also restricted. The population size grows, while viruses disappear.

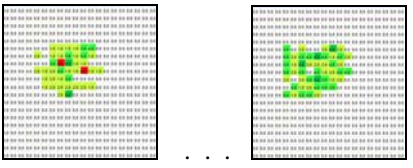


Fig. 2. Congregated population scenario

Since the individuals separate, there is more space to reproduce. However, the number of individuals decreases due to the development of viruses. The number of individuals who survive finally decreases when compared with previous examples:

- population tends to congregate. Reproduction leads to congregation, and congregation is helpful to avoid being killed. However, the introduction of movement has an opposite effect.
- as movement rules have higher priority than reproduction, the possibilities to reproduce are smaller.
- In the previous examples, the initial distribution contained more young individuals. Here, the age is uniformly distributed, and elder individuals die earlier.

Scenario 3. Individuals disperse, viruses scattered. Movement rules are applied.

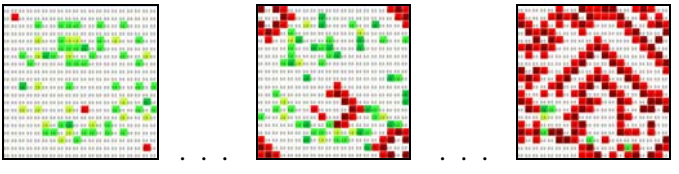


Fig. 3. Movement scenario

3 Conclusions

Cell–DEVS allows describing complex systems using an n-dimensional cell-based formalism. Complex timing behavior for the cells in the space can be defined using very simple constructions. The CD++ tool, based on the formalism entitles the definition of complex cell-shaped models. We have used CD++ to build a Cell-DEVS model on competition between population and viruses. We showed how to define such a model in CD++, and presented different simulation results under different

scenarios. We extended the basic behavior of such model including mobility, showing how to define such a model using CD++ specification facilities.

References

1. Wolfram, S. 2002. A new kind of science. Wolfram Media, Inc.
2. G. Wainer, N. Giambiasi: N-dimensional Cell-DEVS. Discrete Events Systems: Theory and Applications, Kluwer, Vol.12. No.1 (January 2002) 135-157
3. B. Zeigler, T. Kim, H. Praehofer: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press. 2000.
4. G. Wainer: CD++: a toolkit to define discrete-event models. Software, Practice and Experience. Wiley. Vol. 32. No.3. (November 2002) 1261-1306
5. A Cellular Automata Model of Population Infected by Periodic Plague. P.M.A. Sloot, B. Chopard, and A.G. Hoekstra (Eds.): ACRI 2004, LNCS 3305, pp. 464–473, 2004.
6. H. Shang, G. Wainer. A model of virus spreading in CD++. Technical Report SCE-05-05. Systems and Computer Engineering. Carleton University. 2005.

Appendix (Excerpt of the Update Rules for the Model)

```
rule : {~pa:=(0,0)~pa+1[1]; ~pd:=round(uniform[2](1,4));}[3] 100[4]  { (0,0)~pa
=1 and (not ((0,-1)[5]~pa=8 and (0,1)~pa=8)) and not ((-1,0)~pa = 8 and
(1,0)~pa = 8)}[6]  %Age increment for newborns not killed by viruses. rule
: {~pa:=0;} 100 {(0,0)~pa=1 and (((0,-1)~pa=8 and (0,1)~pa=8) or ((-
1,0)~pa=8 and (1,0)~pa=8))}   ; virus killing individuals
%Moving rules for mature cells.
rule : {~pa:=0;} 100 {(0,0)~pa>=2 and (0,0)~pa<=5 and (0,0)~pd=4 and
(0,-1)~pa = 0}
  ...
rule:  {~pa:=0;} 100 {(0,0)~pa=6} % Dying individual
rule: {~pa:=0;} 100 {(0,0)~pa=8 and (((0,-1)~pa>=1 and (0,-1)~pa<=6) or
((0,1)~pa>=1  and  (0,1)~pa<=6)  or  ((-1,0)~pa>=1  and  (-1,0)~pa<=6)  or
((1,0)~pa>=1 and (1,0)~pa<=6))} %Individual kill virus rule
rule:   {~pa:=9;}  100  {(0,0)~pa=8  and  not  (  ((0,-1)~pa>=1  and  (0,-
1)~pa<=6)  or  ((0,1)~pa>=1  and  (0,1)~pa<=6)  or  ((-1,0)~pa>=1  and  (-
1,0)~pa<=6) or ((1,0)~pa>=1 and (1,0)~pa<=6))}
...
rule: {~pa:=0;} 100 {(0,0)~pa=9}
rule: {~pa:=(0,1)~pa+1; ~pd:=round(uniform(1,4));} 100 {(0,0)~pa=0 and
((0,1)~pa>=2 and (0,1)~pa<=5) and (0,1)~pd=4} % Virus: active to passive
%Individual reproduction rule
rule: {~pa:=1;} 100 {(0,0)~pa=0 and ( ((0,1)~pd !=4 or (0,1)~pa<2 or
(0,1)~pa>5) and ((-1,0)~pd !=3 or (-1,0)~pa<2 or (-1,0)~pa>5) and ((0,-
1)~pd !=2 or (0,-1)~pa<2 or (0,-1)~pa>5) and ((1,0)~pd !=1 or (1,0)~pa<2
or (1,0)~pa>5) ) and ((((0,-1)~pa>=2 and (0,-1)~pa<=5) and ((0,1)~pa>=2
```

[1] (0,0)~pa: (0,0) : cell reference, ~pa: associated port.
[2] Random number generator using Uniform distribution.
[3] Postcondition: the corresponding port of reference cell will be updated according this.
[4] Delay time. After the proposed delay time, the output port value of each cell will be updated.
[5] Relative position to the reference cell.
[6] Precondition part. Used to evaluate the current reference cell, if the condition is true, then the cell will be valued according to the value part.

```
and (0,1)~pa<=5) ) or ( ((-1,0)~pa>=2 and (-1,0)~pa<=5) and ((1,0)~pa>=2
and (1,0)~pa<=5)))}
%Virus reproduction rule
rule: {~pa:=8;} 100 {(0,0)~pa=0 and ( ((0,1)~pd !=4 or (0,1)~pa<2 or
(0,1)~pa>5) and ((-1,0)~pd !=3 or (-1,0)~pa<2 or (-1,0)~pa>5) and ((0,-
1)~pd !=2 or (0,-1)~pa<2 or (0,-1)~pa>5) and ((1,0)~pd !=1 or (1,0)~pa<2
or (1,0)~pa>5) ) and not((((0,-1)~pa>=2 and (0,-1)~pa<=5) and
((0,1)~pa>=2 and (0,1)~pa<=5)) or (((-1,0)~pa>=2 and (-1,0)~pa<=5) and
((1,0)~pa>=2 and (1,0)~pa<=5))) and (((0,-1)~pa=8 or (0,1)~pa=8) or ((-
1,0)~pa=8 or (1,0)~pa=8))}
```

A Cellular Automata Model of Competition in Technology Markets with Network Externalities

Judy Frels*, Debra Heisler+, James Reggia+, and Hans-Joachim Schuetze*

* Robert H. Smith School of Business and + Dept. of Computer Science,
University of Maryland, College Park, MD 20742 USA

Abstract. In network markets, the utility a consumer gains from a product increases with the number of other consumers adopting that product. We examine the evolution of such markets using stochastic cellular automata (CA) models to simulate consumer adoption and pricing strategy decisions. Two firms, which lie outside of and interact with the CA model, sponsor incompatible technologies and compete for market dominance using various penetration pricing strategies. We study markets where the two technologies are equal in capability and where they have asymmetric capabilities. We find that the importance of the network to the consumer is critical in determining the financial success of enacting pricing strategies: penetration pricing in a market where network externalities are only moderately important to the consumer can be disastrous. We also find that while an inferior technology may gain substantial market share by price-cutting, it is not likely to gain financial rewards. Finally, we find that if a penetration pricing strategy is to be enacted, more aggressive strategies with larger price cuts lead to greater success.

1 Technology Markets with Network Externalities

Expanding the scientific understanding of collective human behavior and decision making is of interest in many fields, including psychology, sociology, economics, and business management. Cellular automata (CA) models provide a potentially powerful framework for investigating the collective behavior of people, in part due to their explicit representation of space, and their focus on concurrent local computations that lead to self-organization and "emergent behaviors". Limited past work has used CAs to examine such issues as spatial segregation of human populations [1], sales patterns in marketing [2], dissemination of culture [3], evolution of cooperative behavior [4, 5], traffic flow [6], and urban growth patterns [7]. Closely related multi-agent models using simple local interactions in cellular grids have examined issues such as the distribution of wealth [8] and the emergence of communication [9, 10].

In this paper we describe our research using a CA model to investigate human decision making during the adoption of new technology. In many markets, particularly information technology markets, consumers gain benefits from doing what other consumers do, i.e., from adopting compatible products. Such markets are called *network markets* [11, 12] because the value that a consumer gets from a product depends on the network of other consumers associated with that product. For example, tele-

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 378 – 385, 2005.
© Springer-Verlag Berlin Heidelberg 2005

phones and fax machines are valuable only when other people (a "network") have purchased such products. Network markets tend to be "tippy" or "winner-take-all," meaning that as one product gets ahead, it becomes more attractive to the next adopter who, upon adopting it, makes it even more attractive to the next adopter (e.g., VHS vs. Betamax video recorders). Thus, one characteristic of such markets is that often only one technical standard is likely to prevail, and because the value the network provides can overwhelm the value provided by the technology itself, it is possible that the prevailing standard will not be the most technologically advanced.

In such markets, management may adopt strategies to ensure the market tips in their direction rather than their competitor's. Strategies appropriate in network markets often run counter to recommended strategies in non-network markets because it can be critical for a firm to gain an early lead in adoption [11]. Thus, in such a market, it may be worthwhile to pursue strategies that are not immediately revenue enhancing under the assumption that greater revenue can be captured once the market has tipped in the firm's direction. Examples of strategies suggested to managers of network markets include penetration pricing, signaling, product improvements, preannouncements, licensing of the technology, and support for development of complementary products [11, 13]. In this research, we examine the efficacy of a selection of penetration pricing strategies.

2 Methods

We study a CA model of human decision-making in network markets. Our primary goal is to better understand the effects of different pricing strategies that a business can take to promote adoption of its technology where decisions by one consumer affect those of others. Another goal is to assess the effectiveness of CA models as an investigative method for answering real-world questions involving complex socio-economic systems. Our research is conducted in an artificial world we call *StandardScape*, where active cells are viewed as *agents* representing adopters (individual consumers or firms) who choose between two incompatible technological products (A or B) and whose goal is to maximize their utility. We also model technology sponsors representing firms competing to achieve a dominant market share and then recoup their investment in this CA-simulated market. The technology sponsors are external to but interact with and influence the CA model. Each technology sponsor firm has a cost of goods sold, and we track the revenue each firm obtains from the agents adopting its products. If, through the various strategies enacted, the technology sponsors deplete their capital, they go out of business. Their strategy is based on penetration pricing, as follows. When a firm has a market share it does not deem sustainable, it lowers its price to gain further market share. When a firm has a market share that it deems "dominant" and overall market penetration by all providers is significant, it raises its price to recoup earlier revenue foregone by price cutting and to take advantage of its near monopolistic situation [14].

The current set of simulations was done in a 60x60 cellular space having 600 randomly located cells designated to be agents/consumers (see Fig. 1). Agent cells have *multi-attribute states*, i.e., states composed of multiple fields [15]. For example, one field is *Product-Adopted* and can take on values *N* (neither product), *A* (product A

adopted) or *B* (B adopted). All non-agent cells are permanently quiescent. Each agent/consumer cell updates its state based on the state of just those agents (about 18 on average) in its local 11x11 neighborhood. In each time period (conceptualized as a month in this setting), agents seek to adopt a product (A or B) that maximizes their utility. Adopting a product creates network externalities (as well as switching costs) that then influence agents' future decisions. We model a multi-period adoption scenario because repeated adoption is common in business settings [12]; agents may revert from A or B to neither.

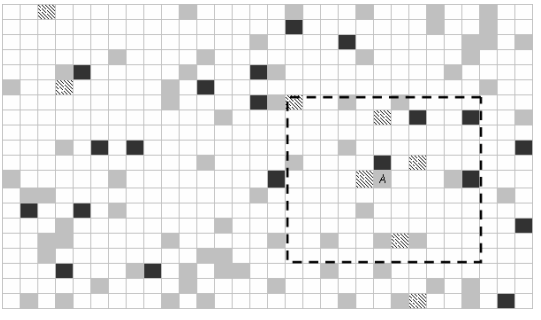


Fig. 1. A portion of the CA space, where quiescent cells are white, and agents adopting product A, B, or neither are shaded cells. The agent labeled "A" here can only observe the state of other agents within its neighborhood (dashed lines)

Cell state changes are determined by a consumer/agent's utility function that is based on a multi-attribute model [16] reflecting the stand-alone technological value of the product, the price of the product, the network of users associated with the product, the consumer's expectations of the future size of the network [11], and the agents' switching costs [17]. In the equation below, *p* designates a particular product, *i* designates an individual consumer agent, and weights $w1_i$—$w5_i$ are positive and sum to 100. Agent *i*'s utility U_{pi} is:

$$U_{pi} = w1_i(Tech_p) - w2_i(Price_p) + w3_i(Network_{pi}) + w4_i(Expect_{pi}) + w5_i(Investment_{pi})$$

where $Tech_p$ represents the performance rating of each of the technologies A and B, $Price_p$ represents the price of each technology (initially \$100 for each), $Network_{pi}$ is calculated as ln(number of neighbors who have bought product p) / ln(total number of neighbors), $Expect_{pi}$ represents agent-specific expectations about the future size of the user network for products A and B [11], and $Investment_{pi}$ represents the investments an agent has made in product A or B to date or their switching costs.

At each time tick, each simulated consumer agent attempts to purchase the technology with the greatest utility provided that the consumer agent has sufficient capital, and that the utility of the chosen product is greater than the reservation utility of *Threshold* (the amount of utility above and beyond the price that a product must provide in order for the consumer to purchase a product rather than choosing "no choice" as a preferred option). Each time period consumer agents are endowed with an incre-

ment of capital ($100) that they are able to spend on technology A or B or save for future spending, depending on their utility function. If they purchase a product, an amount of capital equal to the price of the product (set by the technology sponsor agents) is transferred to the technology sponsor agent as payment. The technology sponsor agents amass capital by a simple calculation of revenue (payments) minus the cost of goods sold ($25 for each item). Summarizing, the state of each agent cell *i* consists of the values of Product-Adopted$_i$, Expect$_{Ai}$, Expect$_{Bi}$, Investment$_{Ai}$, Investment$_{Bi}$, and Capital$_i$, all of which vary during a simulation. Each agent also has *fixed* weights w1$_i$ – w5$_i$ indicating the relative importance of the factors in its utility function. To provide reasonable estimates for these weight values, we collected data from 141 subjects on cell phone feature adoptions that allowed us to estimate their subjective weights using conjoint analysis (space limitations prevent describing this study here). The weights and thresholds used for each of the 600 agent cells in our simulations were randomly sampled, with replacement, from these 141 actual consumers.

Our model is stochastic for several reasons. First, agents are randomly placed. Second, when the product that maximizes the consumer's utility is determined, that product is adopted only with 85% probability to account for events outside the model [18]. Third, expectations regarding the future size of the product networks ($Expect_{pi}$) are randomly assigned to each agent. These values are updated through the duration of the run to reflect the market's evolution. Upon creation, each agent is set to change its expectation every 1 to 12 time ticks. This number is randomly generated and fixed for that agent throughout the run. When it is time for an agent to change its expectation, it will update it based on which technology the majority of its neighbors bought during the last time tick. The expectations are modified by a random number between 0 and 1. The expectation for the technology with the larger market share goes up by the fraction, while the expectation for its competitor goes down by the fraction. The expectations have a minimum of 0 and a maximum of 100.

The two technology sponsors/firms for products A and B are able to take action simultaneously every three time periods (i.e., once per quarter) and react to the market penetration they observe. Each technology sponsor agent has two internal variables: *MktShareTrigger* which describes the market share this sponsor agent wants to achieve once at least 80% of the market has been penetrated; and *PricingFactor* which describes the percentage by which the sponsor will raise or lower the price of the technology. Firms can observe the market share they and their competitors have achieved, as well as overall market penetration. When the market penetration is less than 80%, considered low market penetration, the technology sponsor will lower its price by the PricingFactor in order to attract more buyers. If the market penetration is high (over 80%) the technology sponsor will do one of three things based on its current market share and its MktShareTrigger. When the sponsor's market share is between 20% and its desired market share (MktShareTrigger), the technology sponsor will lower its price by the PricingFactor. When the sponsor's market share is above its desired market share, the technology sponsor will raise its price by the PricingFactor to recoup its earlier investment in obtaining this market share. When the sponsor's market share falls below 20%, the technology sponsor will do nothing in order to retain its loyal customers. Sponsor agents accumulate capital as long as they sell at a price above their costs. Sponsor agents may sell below their cost, but can only operate for 36 time periods (i.e., 36 months, three years, or 12 quarters) with negative

capital. After 36 time periods with negative capital, technology sponsor agents will exit the market (i.e., go out of business.)

Simulations were run in a 5 (pricing strategies) x 2 (importance of network) experimental design. We developed a menu of pricing strategies (MktShareTrigger and PricingFactor) to be enacted by the inferior technology. In all cases except the most basic runs (described below), the inferior technology enacts one of the strategies and the superior technology enacts the least aggressive strategy. We then examine the effectiveness of these strategies in market share and income gained. Market share triggers for price cuts were either 20% or 65%, and penetration pricing factors either 20% or 50%. In the "Most Aggressive" strategy, the technology sponsor will cut price by 50% when its market share drops below 65%. In the "Least Aggressive" strategy, the technology sponsor agent cuts price by 20% when its market share drops below 20%. In the "Itchy but Weak Trigger Finger" (IWTF) strategy, the technology sponsor is quick to cut prices (below 65% market share) but does so by a modest amount (20%). In the "Desperation" strategy, the technology sponsor waits to cut prices until the market share has dropped below 20%, but at that point, cuts them significantly (50%). Each simulation reported below was run 30 times for each experimental condition, so each table entry in the Results that follow is the mean over those 30 runs.

3 Results

The first two rows of Table 1 summarize the results after 5 years when two equal technologies ($Tech_A = Tech_B = 70$) compete in an environment where network externalities are important. Differences in left-to-right-adjacent table entries are statistically significant ($p < .05$) unless marked with an asterisk. As seen here, a more aggressive penetration pricing strategy by technology A leads to larger numbers of adopters and also to greater income. Even when met with meager price-cutting by the competition, this most-aggressive penetration pricing strategy proves successful both financially and from a market share standpoint. This can be contrasted with the outcomes when the network is less important in the consumer agents' utility function (Table 1, last two rows). In these cases, the aggressive network development strategy is somewhat successful from an adoption standpoint, but is always disastrous from a financial standpoint. Further, when the most aggressive strategy is countered by even the smaller price cutting strategy on the competitor's part, both firms lose money. Thus, while A's aggressive pricing strategy consistently gave it a larger market share, it only profited financially when network externalities were important.

Table 1. Equal Technologies, Firm A Implements Most Aggressive Strategy (N=30)

B's Strategy	Network Importance	A's Income	B's Income	Agents Adopting A	Agents Adopting B
No Action	High	176,690	87,245	477.2	9.6
Least Agr.	High	171,085	61,056	422.9	61.5
No Action	Low	-21,307	121,205	444.7	20.1
Least Aggr	Low	-16,880*	-40,261*	290.5	170.6

What if, instead of two equal technologies, one is technically inferior to the other? Table 2 shows the results after 5 years when the firm with the inferior technology (*Tech* = 56) pursues different pricing strategies while the superior technology firm (*Tech* = 70) follows a least-aggressive strategy. The superior technology usually gained a larger market share in all situations except when the inferior technology firm used a most aggressive pricing strategy in a high-importance network market; even then the inferior technology only obtained a marginal market share victory at the expense of a net negative income. The same pattern of results was found when simulations were allowed to continue for over 16 years (data not shown). Thus, our results provide no clear support for believing that inferior technologies can use penetration-pricing strategies effectively to capture the largest market share when agents see only their local neighborhood.

Table 2. Unequal Technologies, Superior Tech. Firm Uses Least Aggressive Strategy

Inf. Tech. Strategy	Network Importance	Inf. Tech. Income	Sup. Tech. Income	Agents Adopting Inf. Tech.	Agents Adopting Sup. Tech.
Base	High	210,290	904,935	47.4	216.6
Least Aggr.	High	18,021	207,037	74.1	405.0
Desperation	High	24,227	166,148	194.3	289.1
IWTF	High	-3,389	254,383	92.4	391.1
Most Aggr.	High	-3,040	180,200	272.7*	207.6*
Base	Low	207,500	518,640	45.8	121.0
Least Aggr.	Low	-8,402	-69,052	89.3	375.5
Desperation	Low	12,748	-85,833	68.4	384.7
IWTF	Low	-6,116	-79,691	77.5	389.3
Most Aggr.	Low	15,566	-81,373	36.8	416.0

Table 2 also indicates that whenever the inferior technology firm undertakes any penetration pricing strategy and the network importance is low, the inferior technology will, on average, receive as much or more income than the superior technology. In contrast, when the network is more important, the superior technology always dominates the inferior in terms of income achieved. The same pattern of results is found when we allow the simulations to run for more than 16 years. Thus, the importance of network externalities to consumers in a market had a major impact on income received.

Finally, we compared the success of strategies enacted by the inferior technology against a superior technology that consistently applied the least aggressive strategy. We found in these and other simulations (data not shown) that both technology sponsors are most financially successful when the inferior technology pursues no penetration pricing at all. On average, the superior technology (enacting a least aggressive strategy) achieves greater income in the base case of no penetration pricing by the inferior technology (mean $711,788) than with any pricing strategy as does the inferior (mean $208,895). Further, the superior technology sees significantly fewer agents adopting in the base scenario compared to all other strategies, an average of 168.8.

The inferior technology see the fewest adopters under the base strategy with an average of 46.6, but this is not significantly lower than in the IWTF (mean 85.0) or Least Aggressive strategies (81.7). The inferior technology obtains the greatest number of agents adopting by using the Most Aggressive strategy (154.8) or Desperation strategy (131.4). The superior technology gains the most agents when the inferior technology employs the Least Aggressive (mean 390.3) or IWTF (mean 390.2) strategy. By the end of year 5, the superior technology sponsor never goes out of business. The inferior technology sponsor agent does go out of business twice when using the Desperation strategy and three times when using the Most Aggressive strategy, both when the network is less important. When we allow simulations to continue for over 16 years, we see much greater frequency of sponsors going out of business. The Least Aggressive and the IWTF (Itchy but Weak Trigger Finger) strategies were most prone to putting the inferior technology sponsor out of business. Both of these strategies involve price cuts of only 20%. In contrast, the superior technology appears to be most likely to go out of business when the inferior technology sponsor adopts the Desperation or Most Aggressive penetration pricing strategy. Both of these strategies involve price cuts of 50%

4 Discussion

Our simulations produced a number of interesting results from a business management perspective. First, the importance of the network to consumers was found to be crucial in determining the success of penetration pricing strategies. A dramatic difference in results occurred when the network was more heavily weighted in the consumer's utility function versus when it was not. Thus, misinterpreting the relevance of network externalities to a firm's potential market, although it may lead to successful market share development, may also lead to disastrous financial results. Also, when technologies are symmetric and when a firm has "lost" the market share battle, little was to be gained by even mild price-cutting. Second, when technologies are asymmetric, an inferior technology sponsor enacting a penetration pricing strategy may occasionally achieve market share success but that success is not necessarily tied to financial success. Finally, we observed that no penetration pricing at all brings the greatest financial reward to the sponsor agents, although it achieves very low market penetration. It is possible that this just reflects the manner in which we implemented the pricing rule; it is probably more reasonable to compare outcomes of different strategies to one another given that they are all implemented with the same biases. Within the strategies, we found that strategies with larger price reductions led to the largest number of adopters for the inferior technology. These same strategies, enacted by the inferior technology sponsor, are also most damaging to the superior technology's financial and market share position. Despite the market share results achieved with the Desperation and Most Aggressive strategies, such price cuts did not necessarily lead to better or worse financial success for the inferior technology. However, in terms of punishing a competitor with a superior technology our results suggest the value of true penetration pricing rather than modest price cuts in a network market.

Acknowledgement. Supported by Marketing Science Institute research grant #4-1235 and by the Robert H. Smith School of Business Summer Research Fund; JR was supported by NSF award IIS-0325098.

References

1. Schelling T. Models of Segregation, *American Economic Review*, 59, 1969, 488-493.
2. Goldenberg J., Libai B. and Muller E. Riding the Saddle: How Cross-Market Communications can Create a Major Slump in Sales, *Journal of Marketing*, 66, 2002, 1 - 16.
3. Axelrod, R. (1997), The Dissemination of Culture, *Journal of Conflict Resolution*, 41, 203-226.
4. Nowak M. and May R. (1992) Evolutionary Games and Spatial Chaos, *Nature*, 359, 826-829.
5. Hauert C. and Doebeli M. (2004), Spatial Structure Often Inhibits the Evolution of Cooperation in the Snowdrift Game, *Nature*, 428, 643-646.
6. Gaylord R. and Nishidate K. (1996), *Modeling Nature*, Springer, pp. 25-35.
7. Back T., et al. (1996), *Modeling Urban Growth by Cellular Automata, Parallel Problem Solving from Nature IV*, Springer, 636-645.
8. Epstein J. and Axtell R. *Growing Artificial Societies*, MIT Press, 1996.
9. Reggia, J., Schulz R., Wilkinson G., and Uriagereka J. (2001), Conditions Enabling the Evolution of Inter-Agent Signaling in an Artificial World, *Artificial Life*, 7, 3-32.
10. Wagner K., Reggia J., Uriagereka J .and Wilkinson G. Progress in the Simulation of Emergent Communication and Language, *Adaptive Behavior*, 11, 2003, 37-69.
11. Besen, S. and Farrell J. (1994), Choosing How to Compete: Strategies and Tactics in Standardization, *Journal of Economic Perspectives*, 8(2), 117-131.
12. Frels, J,, Shervani T., and Srivastava R. (2003), The Integrated Networks Model: Explaining Resource Allocations in Network Markets, *Journal of Marketing*, 67(1), 29-45.
13. Hill, C. (1997), Establishing a Standard: Competitive Strategy and Technological Standards in Winner-Take-All Industries, *Academy of Management Exec.* 11, 1997, 7-25.
14. Farrell, J. and Saloner G (1986), Installed Base and Compatibility: Innovation, Product Preannouncements, and Predation, *The American Economic Review*, 76(5), 940-955.
15. Chou H. & Reggia J (1997). Emergence of Self-replicating Structures in a Cellular Automata Space, *Physica D,* 110, 252-276.
16. Fishbein, M. and Icek A. (1975), *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research,* Reading, Mass. : Addison-Wesley Pub. Co.
17. Burnham, T. Frels J., Mahajan V. (2003), Consumer Switching Costs *Journal of the Academy of Marketing Science*, 109-126.
18. Arthur, W. (1989), Competing Technologies, Increasing Returns, and Lock-in by Historical Events, *The Economic Journal*, 99 (March), 116-131.

Self-organizing Dynamics for Optimization

Stefan Boettcher

Physics Department, Emory University, Atlanta, Georgia 30322-2430, USA

Abstract. Motivated by noise-driven cellular automata models of self-organized criticality (SOC), a new paradigm for the treatment of hard combinatorial optimization problems is proposed. An extremal selection process preferentially advances variables in a poor local state. The ensuing dynamic process creates broad fluctuations to explore energy landscapes widely, with frequent returns to near-optimal configurations.

1 Motivation from Self-organized Criticality

The study of certain randomly-driven, dissipative automata model with local threshold dynamics has provided a plausible view of many self-organizing processes ubiquitous in Nature [1]. Most famously, the Abelian sandpile model [3] has been used to describe the statistics of earthquakes [1]. Another variant is the Bak-Sneppen model (BS) [2], in which variables are updated sequentially based on a global threshold condition. It provides an explanation for broadly distributed extinction events [21] and the "missing link" problem [14]. Complexity in these SOC models *emerges* purely from the dynamics, *without* fine-tuning of parameters, as long as driving is slow and the ensuing avalanches are fast.

In the BS, "species" are located on the sites of a lattice, and have an associated "fitness" value between 0 and 1. At each time step, the one species with the smallest value (poorest degree of adaptation) is selected for a random update, having its fitness replaced by a new value drawn randomly from a flat distribution on the interval $[0, 1]$. But the change in fitness of one species impacts the fitness of interrelated species. Therefore, all of the species at neighboring lattice sites have their fitness replaced with new random numbers as well. After a sufficient number of steps, the system reaches a highly correlated state known as self-organized criticality (SOC) [3]. In that state, almost all species have reached a fitness above a certain threshold. These species, however, possess *punctuated equilibrium* [14]: only one's weakened neighbor can undermine one's own fitness. This coevolutionary activity gives rise to chain reactions called "avalanches", large fluctuations that rearrange major parts of the system, potentially making any configuration accessible.

We have described [7] the spatio-temporal complexity of the BS by a *delay* equation for the evolution of activity $P(r, t)$ after a perturbation at $(r = 0, t = 0)$,

$$\partial_t P(r,t) = \nabla_r^2 P(r,t) + \int_{t'=0}^{t} dt' V(t-t') P(r,t'), \qquad V(t) \sim t^{-\alpha}, \tag{1}$$

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 386–394, 2005.
© Springer-Verlag Berlin Heidelberg 2005

which has the solution $P(r,t) \sim \exp\{-C(r^D/t)^{\frac{1}{D-1}}\}$ with $D = 2/(\alpha - 1)$. Thus, it is the *memory* of all previous events that determines the current activity.

2 Extremal Optimization

Although coevolution may not have optimization as its exclusive goal, it serves as a powerful paradigm. We have used it as motivation for a new approach to approximate hard optimization problems [8, 9]. The heuristic we have introduced, called extremal optimization (EO), follows the spirit of the BS, updating those variables which have among the "worst" values in a solution and replacing them by random values without ever explicitly improving them. The resulting heuristic explores the configuration space Ω widely with frequent returns to near-optimal solutions, *without* fine-tuning of parameters.

To introduce EO, let us consider a spin glass [20] as a specific example of an NP-hard optimization problem. It consists of a d-dimensional hyper-cubic lattice of length L with periodic boundary conditions, with a spin variable $x_i \in \{-1, 1\}$ at each site i, $1 \leq i \leq n$ $(= L^d)$. A spin is connected to each of its nearest neighbors j via a bond variable $J_{ij} \in \{-1, 1\}$, assigned at random. We minimize

$$C(S) = H(\mathbf{x}) = -\sum\nolimits_{\langle i,j \rangle} J_{ij} x_i x_j \tag{2}$$

as (Hamiltonian) cost function, the sum extending over all nearest-neighbor pairs of spins. Due to frustration [20], ground states $S_{\min}$ are hard to find for $d > 2$.

To find near-optimal solutions for a particular optimization problem, EO performs a neighborhood search on a single configuration $S \in \Omega$. As in the spin problem in Eq. (2), S consists of a large number n of variables x_i. We assume that each S possesses a neighborhood $N(S)$ that rearranges the state of merely a small number of the variables. This is a characteristic of a local search, in contrast to a genetic algorithm, say, where cross-overs may effect $O(n)$ variables on each update. The cost $C(S)$ is assumed to consist of the individual cost contributions, or "fitnesses", λ_i for each variable x_i. The fitness of each variable assesses its contribution to the total cost and typically the fitness λ_i depends on the state of x_i in relation to connected variables. For example, for the Hamiltonian in Eq. (2), we assign to each spin x_i the fitness

$$\lambda_i = x_i \sum_{<,j>} J_{ij} x_j, \qquad C(S) = -\sum_{i=1}^{n} \lambda_i. \tag{3}$$

Each spin's fitness thus corresponds to (the negative of) its local energy contribution to the overall energy of the system. In similarity to the BS, EO then proceeds through a neighborhood search of Ω by sequentially changing variables with "bad" fitness on each update, for instance, via single spin-flips. After each update, the fitnesses of the changed variable and of all its connected neighbors are reevaluated according to Eq. (3).

The algorithm operates on a single configuration S at each step. Each variable x_i in S has a fitness, of which the "worst" is identified. This *ranking* of the

variables provides the only measure of quality on S, implying that all other variables are "better" in the current S. In the move to a neighboring configuration, typically only a small number of variables change state, so only a few connected variables need to be re-evaluated [step (2a)] and re-ranked [step (2b)]. In detail:

1. Initialize configuration S at will; set $S_{\rm best} := S$.
2. For the "current" configuration S,
   (a) evaluate λ_i for each variable x_i,
   (b) find j satisfying $\lambda_j \leq \lambda_i$ for all i, *i.e.*, x_j has the "worst fitness",
   (c) choose $S' \in N(S)$ such that x_j *must* change,
   (d) accept $S := S'$ *unconditionally,*
   (e) if $C(S) < C(S_{\rm best})$ then set $S_{\rm best} := S$.
3. Repeat at step (2) as long as desired.
4. Return $S_{\rm best}$ and $C(S_{\rm best})$.

There is no parameter to adjust for the selection of better solutions. It is the *memory* encapsulated in the ranking that directs EO into the neighborhood of increasingly better solutions. Like BS, those "better" variables possess punctuated equilibrium: their memory only get erased when they happen to be connected to one of the variables forced to change. On the other hand, in the choice of move to S', there is no consideration given to the outcome of such a move, and not even the worst variable x_j itself is guaranteed to improve its fitness. Accordingly, large fluctuations in the cost can accumulate in a sequence of updates. Merely the bias *against* extremely "bad" fitnesses produces improved solutions.

Tests have shown that this basic algorithm is very competitive for optimization problems [8]. But in cases such as the single spin-flip neighborhood for the spin Hamiltonian, focusing on only the worst fitness [step (2b)] leads to a deterministic process, leaving no choice in step (2c): If the "worst" spin x_j *has* to flip and any neighbor S' differs by only one flipped spin from S, it must be $S' = (S/x_j) \cup \{-x_j\}$. This deterministic process inevitably will get stuck near some poor local minimum. To avoid these "dead ends" and to improve results [8], we introduce a single parameter into the algorithm. Ranking all x_i according to fitness λ_i, *i.e.*, we find a permutation Π of the variable labels i with

$$\lambda_{\Pi(1)} \leq \lambda_{\Pi(2)} \leq \ldots \leq \lambda_{\Pi(n)}. \tag{4}$$

The worst variable x_j [step (2b)] is of rank 1, $j = \Pi(1)$, and the best variable is of rank n. Now, consider a scale-free probability distribution over the *ranks* k,

$$P_k \propto k^{-\tau}, \qquad 1 \leq k \leq n, \tag{5}$$

for a fixed value of the parameter τ. At each update, select a rank k according to P_k. Then, modify step (2c) so that x_j with $j = \Pi(k)$ changes its state.

For $\tau = 0$, this "τ-EO" algorithm is simply a random walk through Ω. Conversely, for $\tau \to \infty$, it approaches a deterministic local search, only updating the lowest-ranked variable, and is bound to reach a dead end (see Fig. 1). However,

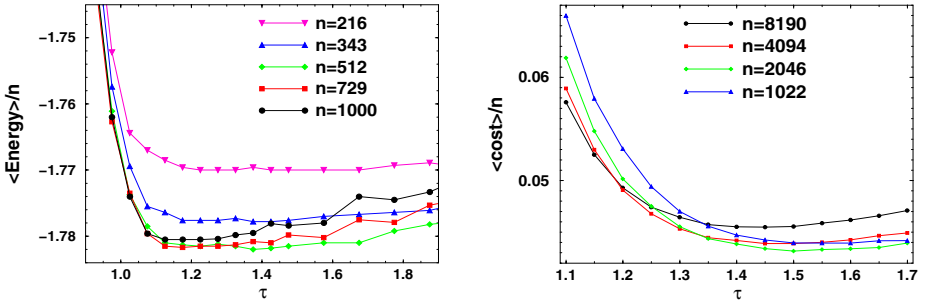


Fig. 1. Plot of costs obtained by EO for a $\pm J$ spin glass (left) and for graph bipartitioning (right), both as a function of τ. For each size n, a number of instances were generated. For each instance, 10 different EO runs were performed at each τ. The results were averaged over runs and instances. Although both problems are quite distinct, in either case the best results are obtained at a value of τ with $\tau \to 1^+$ for $n \to \infty$

for finite values of τ the choice of a *scale-free* distribution for P_k in Eq. (5) ensures that no rank gets excluded from further evolution while maintaining a bias against variables with bad fitness. In all problems studied, a value of

$$\tau - 1 \sim 1/\ln n \qquad (n \to \infty) \tag{6}$$

seems to work best [9, 10]. We have studied a simple model problem for which the asymptotic behavior of τ-EO can be solved exactly [6]. The model reproduces Eq. (6) exactly in cases where the model develops a "jam" amongst its variables, which is quite a generic feature of frustrated systems.

In Fig. 2 we show the range of states that are sampled during a typical run of EO, here for a spin-glass instance with $n = 7^3$ and for the image alignment problem [19]. Starting with a random initial condition, for the first $O(n)$ update steps EO establishes local order, leading to a rapid decrease in the energy. After

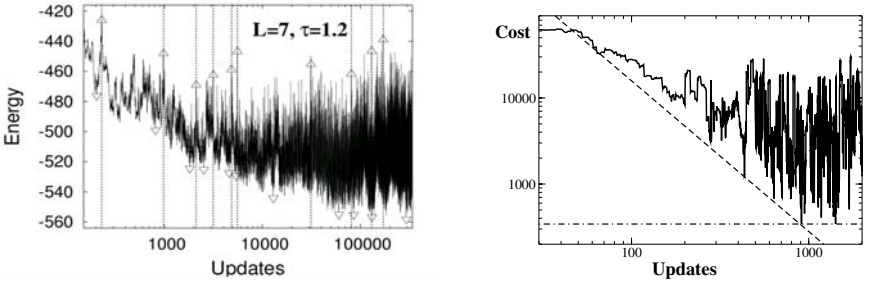


Fig. 2. Plots of the range of states attained by EO during single run on a particular instance (of an $L = 7$ cubic spin glass (left) and of the image alignment problem [19] (right). After an initial transient, the ultimate "steady state" is reached in which EO fluctuates widely through near-optimal configurations, obtaining increasingly better energy records ($\bigtriangledown$) while scaling ever higher barriers ($\bigtriangleup$)

that EO searches through a wide band of states with frequent returns to near-optimal configurations.

3 Numerical Results for EO

In the few years since we first proposed (EO) as a general purpose heuristic for some of the hardest combinatorial optimization problems [8], ample evidence has been provided for its practicality [9, 10, 11]. Our own studies have focused on demonstrating elementary properties of EO in a number of implementations for classic NP-hard combinatorial problems such as graph bipartitioning [8, 10], 3-coloring [11], spin glasses [9], and the traveling salesperson [8]. Several other researchers have picked up on our initial results, and have successfully applied EO to problems as diverse as pattern recognition [19], signal filtering of EEG noise [24], artificial intelligence [18], and $3d$ spin-glass models [12, 23]. Comparative studies have shown that EO holds significant promise to provide a new, *alternative* approach to approximate many intractable problems [8, 12, 18].

3.1 Results on Spin Glasses

To gauge τ-EO's performance for larger $3d$-lattices, we have run our implementation also on two instances, *toruspm*3-8-50 and *toruspm*3-15-50, with $n = 512$ and $n = 3375$, considered in the $7th$ DIMACS challenge for semi-definite problems[1]. The best available bounds (thanks to F. Liers) established for the larger instance are $H_{\text{lower}} = -6138.02$ (from semi-definite programming) and $H_{\text{upper}} = -5831$ (from branch-and-cut). EO found $H_{\text{EO}} = -6049$ (or $H/n = -1.7923$), a significant improvement on the upper bound and already lower than $\lim_{n\to\infty} H/n \approx -1.786\ldots$ found in Ref. [9]. Furthermore, we collected 10^5 such states, which roughly segregate into three clusters with a mutual Hamming distance of at least 100 distinct spins; though at best a small sample of the $\approx 10^{73}$ ground states expected [15]! For the smaller instance the bounds given are -922 and -912, while EO finds -916 (or $H/n = -1.7891$) and was terminated after finding 10^5 such states. While this run (including sampling degenerate states) took only a few minutes of CPU (at 800 MHz), the results for the larger instance required about 16 hours.

More recently, we have combined EO with reduction methods for sparse graphs [4, 5]. These reductions strip graphs of all low-connected variables ($\alpha \leq 3$), thereby eliminating many entropic barriers that tend to bog down local searches [22]. Along the way, the rules allow for an accounting of the *exact* ground-state energy and entropy, and even of the approximate overlap distribution [5]. The "remainder" graph is subsequently handled efficiently with EO. With such a meta-heuristic approach, for example, we have been able to determine the defect energy distribution [13] for $d = 3, \ldots, 7$ dimensional spin

[1] http://dimacs.rutgers.edu/Challenges/Seventh/

glasses, bond-diluted to just above their percolation point, with great accuracy for lattices up to $L = 30$ [4]. As one result, we reduced the error on the stiffness exponent in $d = 3$, $y_{d=3} = 0.240(5)$, from 20% to about 2%. This fundamental exponent describes the energetic cost ΔE of perturbations (here, induced interfaces) of size L, $\sigma(\Delta E) \sim L^y$.

Currently, we are using this meta-heuristic to explore the (possible) onset of replica symmetry breaking (RSB) for sparse mean-field and lattice models just above percolation. So far, we have only some preliminary data for Spin glasses on random graphs. In this model at connectivities near percolation $\alpha \approx \alpha_p = 1$, many spins may be entirely unconnected while a finite fraction is sufficiently connected to form a "giant component" in which interconnected spins may become overconstrained. There the reduction rules allow us to *reduce completely* a statistically significant number of graphs with up to $n = 2^{18}$ spins even well above α_p, since even higher-connected spins may become reducible eventually after totally reducible substructures (trees, loops, etc) emanating from them have been eliminated. At the highest connectivities reached, even graphs originally of $n = 2^{18}$ had collapsed to at most 100 irreducible spins, which EO easily optimized.

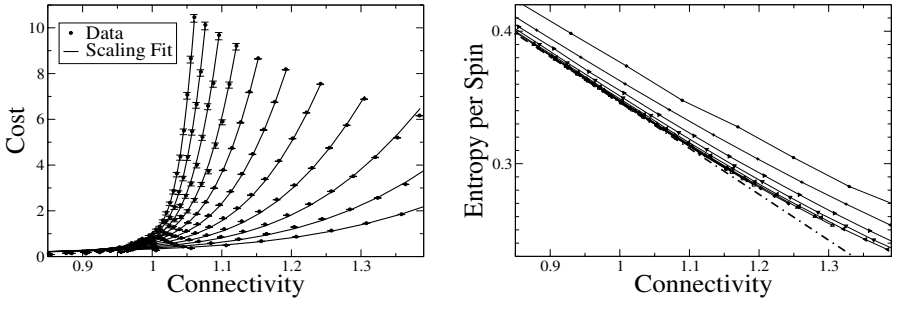


Fig. 3. Plot (left) of the cost and (right) of the entropy per spin, as a function of connectivity α for random graphs of size $n = 2^8, 2^9, \ldots, 2^{18}$. For increasing n, the cost approaches a singularity at $\alpha_{\rm crit} = 1.003(9)$, as determined from a finite-size scaling fit (lines on left) to $\langle C \rangle(\alpha, n) \sim n^{\delta} f\left[(\alpha - \alpha_{\rm crit})\, n^{1/\nu}\right]$. The fit predicts also $\delta = 0.11(2)$ and $\nu = 3.0(1)$. The entropy per spin quickly converges to $\approx (1 - \alpha/2) \ln 2$ (dashed line), exact for $\alpha < \alpha_{\rm crit} = 1$, continues unaffected through the transition, but deviates from that line for $\alpha > \alpha_{\rm crit}$

As a result, we have measured the cost of ground states, Eq. (2), as a function of connectivity α on 40 000 instances for each size $n = 2^8, 2^9, \ldots, 2^{14}$, and 400 instances for $n = 2^{15}, \ldots, 2^{18}$, at each of 20 different connectivities α as shown in Figure 3. We also account *exactly* for the degeneracy of each instance, which could number up to $\exp[0.3 \times 2^{18}]$; minuscule compared to all $2^{2^{18}}$ configurations! Not entirely reduced graphs had their entropy determined with EO in our meta-heuristic. Consistent with theory [17], Figure 3 shows that the entropy per spin follows $s \approx (1 - \alpha/2) \ln 2$ for $\alpha < \alpha_{\rm crit} = 1$, then continues smoothly

through the transition but deviates from that line for $\alpha > \alpha_{\rm crit}$. Similar data for the overlap-moments [5] may determine the onset of RSB expected for this model.

3.2 Applications of EO by Others

The generality of the EO method beyond the domain of spin-glass problems has recently been demonstrated by Meshoul and Batouche [19] who used the EO algorithm as described above successfully on a standard cost function for aligning natural images. Fig. 4 demonstrates the results of their implementation of τ-EO for this pattern recognition problem. Here, τ-EO finds an optimal affine transformation between a target image and its reference image using a set of n *adjustable* reference points which try to attach to characteristic features of the target image. The crucial role played by EO's *non-equilibrium fluctuations* in the local search is demonstrated in Fig. 2. The fluctuations in the image alignment problem are amazingly similar to those we have found for spin glasses. As our discussion in Sec. 2 suggests, they are one of the key distinguishing features of EO, and are especially relevant for optimizing highly disordered systems. For instance, Dall and Sibani [12] have observed a significantly broader distribution of states visited – and thus, better solutions found – by τ-EO compared to simulated annealing [16] when applied to the Gaussian spin-glass problem.

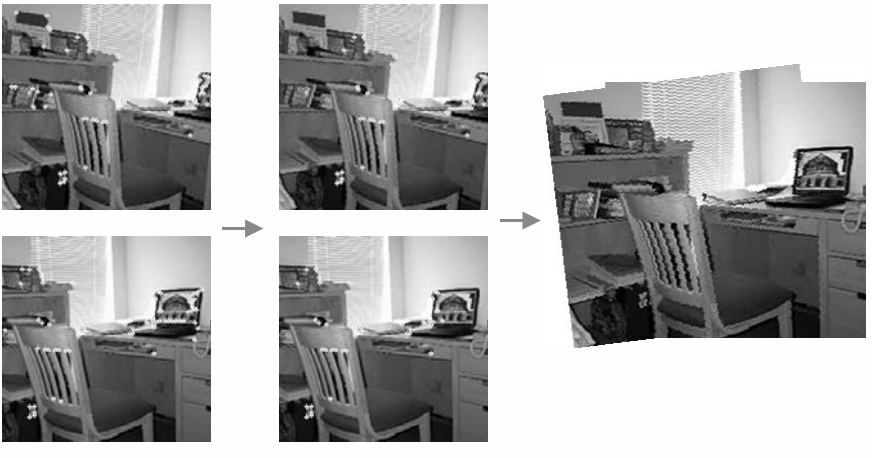


Fig. 4. Application of EO to the image matching problem, after [19]. Two different images of the same scene (top row and bottom row) are characterized by a set of n points assigned by a standard pattern recognition algorithm. Starting from an initial assignment (left, top and bottom), the points are updated according to EO, see also Fig. 2, leading to an optimal assignment (center, top and bottom). This optimal assignment minimizes a cost function for the affine transformation, facilitating an automated alignment of the two images (right). Note that the points move to the part of the scene for which both images overlap. Special thanks to M. Batouche for providing those images

Acknowledgements

I would like to thank M. Paczuski, A.G. Percus, and M. Grigni for their collaboration on many aspects of the work presented here. This work was supported under NSF grant DMR-0312510 and Emory's URC.

References

1. P. Bak, *How Nature Works* (Springer, New York, 1996).
2. P. Bak and K. Sneppen, *Punctuated Equilibrium and Criticality in a simple Model of Evolution,* Phys. Rev. Lett. **71**, 4083-4086 (1993).
3. P. Bak, C. Tang, and K. Wiesenfeld, *Self-Organized Criticality,* Phys. Rev. Lett. **59**, 381 (1987).
4. S. Boettcher, *Low-Temperature Excitations of Dilute Lattice Spin Glasses,* Europhys. Lett. **67**, 453-459 (2004).
5. S. Boettcher, *Reduction of Spin Glasses applied to the Migdal-Kadanoff Hierarchical Lattice,* Euro. Phys. J. B **33**, 439-445 (2003).
6. S. Boettcher and M. Grigni, *Jamming Model for the Extremal Optimization Heuristic,* J. Math. Phys. A: Math. Gen. **35**, 1109-1123 (2002).
7. S. Boettcher and M. Paczuski, *Ultrametricity and Memory in a Solvable Model of Self-Organized Criticality,* Physical Review E **54**, 1082 (1996).
8. S. Boettcher and A. G. Percus, *Nature's Way of Optimizing,* Artificial Intelligence **119**, 275-286 (2000).
9. S. Boettcher and A. G. Percus, *Optimization with Extremal Dynamics,* Phys. Rev. Lett. **86**, 5211-5214 (2001).
10. S. Boettcher and A. G. Percus, *Extremal Optimization for Graph Partitioning,* Phys. Rev. E **64**, 026114 (2001).
11. S. Boettcher and A. G. Percus, *Extremal Optimization at the Phase Transition of the 3-Coloring Problem,* Physical Review E **69**, 066703 (2004).
12. J. Dall and P. Sibani, *Faster Monte Carlo Simulations at Low Temperatures: The Waiting Time Method,* Computer Physics Communication **141**, 260-267 (2001).
13. K. H. Fischer and J. A. Hertz, *Spin Glasses* (Cambridge University Press, Cambridge, 1991).
14. S. J. Gould and N. Eldridge, *Punctuated Equilibria: The Tempo and Mode of Evolution Reconsidered,* Paleobiology **3**, 115-151 (1977).
15. A. K. Hartmann, *Ground-state clusters of two-, three- and four-dimensional +-J Ising spin glasses,* Phys. Rev. E **63**, 016106 (2001).
16. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing,* Science **220**, 671-680 (1983).
17. M. Leone, F. Ricci-Tersenghi, and R. Zecchina, *Phase coexistence and finite-size scaling in random combinatorial problems,* J. Phys. A. **34**, 4615 (2001).
18. M. B. Menai and M. Batouche, *Approximate solution of Max-SAT problem using Extremal Optimization heuristic,* Journal of Automated Reasoning, (to appear).
19. S. Meshoul and M. Batouche, *Robust Point Correspondence for Image Registration using Optimization with Extremal Dynamics,* Lect. Notes Comput. Sc. **2449**, 330-337 (2002).
20. M. Mezard, G. Parisi, and M. A. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore, 1987).

21. D. M. Raup and J. J. Sepkoski, *Periodic Extinction of Families and Genera,* Science **231**, 833-836.
22. F. Ricci-Tersenghi, M. Weigt, and R. Zecchina, *Simplest random K-satisfiability problem,* Phys. Rev. E **63**, 026702 (2001).
23. J.-S. Wang and Y. Okabe, *A comparison of extremal optimization with flat-histogram dynamics for finding spin-glass ground states,* J. Phys. Soc. Jpn. **72**, 1380-1383 (2003).
24. E. Yom-Tov, A. Grossman, and G. F. Inbar, *Movement-related potentials during the performance of a motor task I: The effect of learning and force,* Bio. Cybernatics **85**, 395-399 (2001).

Constructibility of Signal-Crossing Solutions in von Neumann 29-State Cellular Automata

William R. Buckley[1] and Amar Mukherjee[2]

[1] California Evolution Institute, San Francisco, CA. 94134
wrb@calevinst.org
[2] Professor of Computer Science, School of Computer Science,
University of Central Florida, Orlando, FL. 32816
amar@cs.ucf.edu

Abstract. In von Neumann 29-state cellular automata, the crossing of signals is an important problem, with three solutions reported in the literature. These solutions greatly impact automaton design, especially self-replicators. This paper examines these solutions, with emphasis upon their constructibility. We show that two of these solutions are difficult to construct, and offer an improved design technique. We also argue that solutions to the signal-crossing problem have implications for machine models of biological development, especially with regard to the cell cycle.

1 Von Neumann 29-State Cellular Automata Signal-Crossing

John von Neumann developed cellular automata theory, yielding an environment in which to demonstrate his thesis that machines may be designed having the property of self-replication [1]. Von Neumann cellular automata are characterized by a two-dimensional, rectilinear lattice network of finite state automata (the cells), each identical in form, function, and association, as specified by a set of states, a set of rules for the transition of cells between states (the state transition function), and a grouping function that places each cell at the center of a neighborhood of adjacent cells (specifying the set of cells operated upon by the state transition function in the computation of state transitions). All cells transition their state synchronously.

States are grouped into five categories; a ground state, the transition states, the confluent states (C), the ordinary transmission states (→), and the special transmission states (⇑). The last three categories have an activity property, while the last two categories have the property of direction. Activity corresponds to carried data, it being transmitted between states at the rate of one bit per application of the state transition function. Confluent states have the additional property of a one-cycle delay, and so hold two bits of data. The direction property indicates the flow of data between states. Ordinary and special transmission states have an antagonistic relationship, with mutually directed active cells of each causing the annihilation of the other, to yield the ground state. Active special transmission states also yield confluent state annihilation. Confluent states accept data from ordinary transmission states, perform a logical AND on

[1] AKA - Amar Mukhopadhyay

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 395 – 403, 2005.
© Springer-Verlag Berlin Heidelberg 2005

the inputs, and transmit data to both ordinary and special transmission states. Ordinary and special transmission states logically OR inputs. An ordinary transmission state accepts input only from like states, and from adjacent confluent states. Special transmission states accept input likewise. Confluent states pass data to any adjacent transmission state not pointed at the confluent state. Data are not transmitted to transmission states against the direction of those transmission states. For instance, two ordinary transmission states pointing at each other do not exchange data. Instead, the data is simply lost. Data held by a confluent state is lost if there is no adjacent transmission state not pointing at the confluent state.

Patterns of cells are called configurations, with those that implement specific functionality being called organs. Configurations can be compared in terms of their constructibility. Constructibility is both an absolute measure, and a relative measure. Some configurations are not constructible, while other configurations are constructible. In absolute terms, constructibility is the property that a configuration can be obtained through the act of another configuration. In relative terms, constructibility is an inverse measure of effort. In von Neumann 29-state cellular automata, the organ that facilitates configuration construction is known as the construction arm.

2 The Signal-Crossing Problem and Available Solutions

A problem arises within any two-dimensional system respecting the mechanisms of translocation - the crossing problem. The familiar example is roadway transportation, the solutions being stop-and-go intersections, bridges, and traffic circles. In cellular automata, we have the signal-crossing problem. This owes to the fixed-position nature of the component finite state automata, where the translocation is of data (in the form of signals). In such cases, translocation is called communication.

Signals are an ordered sequence of data (bits), whether of fixed or arbitrary length, that are communicated between organs. The literature reports three solutions to the signal-crossing problem within von Neumann 29-state cellular automata. These signal-crossing organs are the Coded Channel (CC), the Mukhopadhyay Crossing Organ (MCO) [2], and the Real-Time Crossing Organ (RTCO). We are here concerned with the properties of these signal-crossing organs, particularly the latter two.

The MCO and RTCO are general signal-crossing solutions, able to serve the crossing needs of any two signals, regardless of length. The CC is a more constrained signal-crossing solution, capable of serving only signals of varying fixed length, though extendable to service an arbitrarily large number of signals. While the MCO and RTCO are indiscriminate in the signals they service, the CC discriminates between signals, via selective acceptance. The function of the CC is programmable, while neither the MCO nor the RTCO is programmable. We now consider signal-crossing organ architecture [3].

The CC has two-layers, with an internal channel (or signal path) positioned between inputs and outputs. The internal channel is of finite length, and is non-cyclic. The first CC layer accepts signal input and translates it into a code carried by the internal

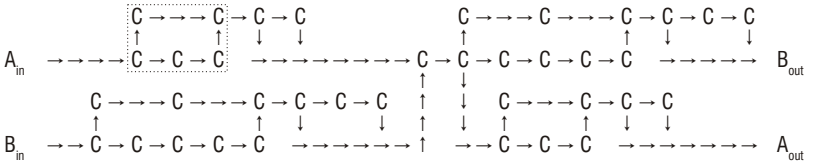


Fig. 1. The minimal CC is a configuration that crosses two signals, <11> and <101>. Input A_{in} is accepted by a decoder/pulser pair, the result being then injected into the internal channel, where an identical decoder/pulser pair again accepts the signal. A single ordinary transmission state separates the decoder from the pulser, each organ being constructed of confluent and ordinary transmission states. The decoder of input A_{in}, outlined in this figure with dashed lines, is an organ of dimension five cells by three cells

channel. The second layer of the CC translates this code into signal output. The CC may accept any input signal a multiple number of times, and may generate the corresponding output signal any number of times. Linearity of the internal channel requires input acceptance prior to output generation. Each input may accept more than one signal, while each output generates only one signal. If corruption of channel code occurs, unwanted output signal generation may result. Thus, signal inputs ought occur with sufficient relative delay. The CC is best applied where it is known that signals are incident only upon the complete servicing of any previously accepted signal.

In the simplest implementation, shown in figure 1, the CC expresses a bijection of two inputs to two outputs. It is especially easy to see in this case that signal input can be both crossed and duplicated on output. If the input signals *A* and *B* are accepted in that order, with input acceptors coming before output generators, and the order of outputs is *B* then *A*, we have that the signals are crossed and duplicated. For signals <11> and <101>, the CC covers approximately 230 cells. CC size is proportional to the number and length of inputs and outputs.

The RTCO is a square organ, comprising 64 cells, as shown in figure 2. It has two inputs and two outputs, arranged in orthogonal input/output pairs. Signals are duplicated at input, routed along a pair of internal paths, and joined into a single signal at output. There are four different signal paths internal to the RTCO, all of identical length.

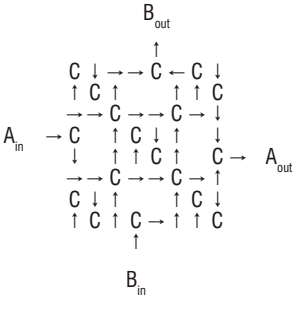


Fig. 2. The RTCO, shown without clock signals, with inputs and outputs indicated

The RTCO has five clocks, each of identical structure and emitting a period six signal <101010>, which drive inputs to outputs and operate in-phase with one-another. Four of these clocks are positioned at the four corners of the RTCO, with the fifth clock located directly in the middle of the RTCO. The four internal signal paths of the RTCO completely surround the central clock. Every other bit of signal input is transmitted along one internal path of the pair while the alternating bits are transmitted along the other path of the pair. Signal transmission through the RTCO is facilitated by the alternating signals generated by the component clocks, which trigger confluent states along the internal paths. These confluent states act as gates to control signal propagation. There are four such gates, each intersecting two internal paths.

Like the RTCO, the MCO has two inputs and two outputs, and the functional elements of the internal paths are similarly co-linear. Unlike the RTCO, the inputs and outputs of the MCO are parallel. The MCO is highly compartmentalised, with a greater variety of functional parts, and exhibits several levels of structure. At the macro-level, the MCO is composed of three units that implement the logical XOR operator, one upstream, and two downstream. The two signals to be crossed are routed through the upstream XOR, the output of which is then routed through the downstream XOR units, each taking as the other input the alternate of the two signals to be crossed. The outputs of these two downstream XOR units will be the two input signals to the MCO, now crossed. A single XOR is shown in figure 3.

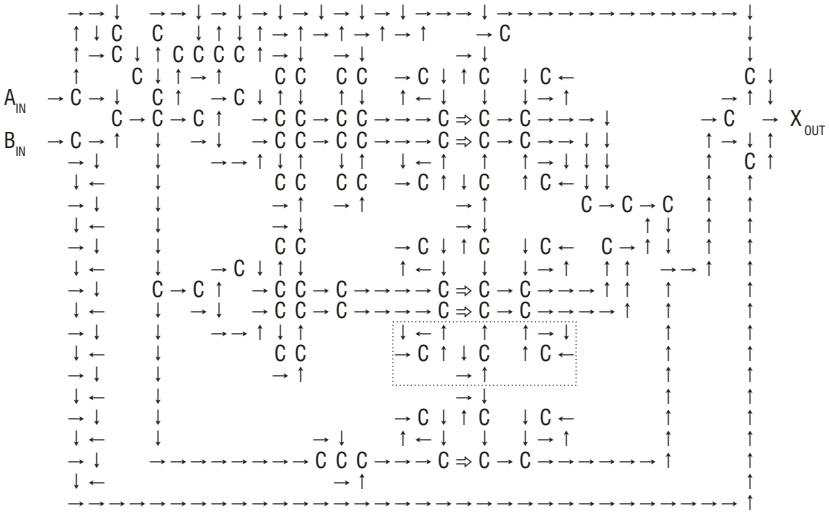


Fig. 3. A single XOR unit of the MCO, shown without clock signals. The five internal paths are clearly visible, as indicated by presence of special transmission states. The clocks of the selector, inverter, collector combination of one path are outlined with a dashed box. The confluent state to the left of the special transmission state is the operational part of the selector, while the confluent state to the right is the operational part of the inverter. The confluent state still further to the right is the operational part of the collector. This organ cannot be autoinitialised

At the meso-level, each XOR of the MCO is composed of three operational parts, two logical AND operators and a logical negation (NOT) operator, and three internal paths. Each of the two signal inputs to the XOR is first duplicated. One of the duplications of each signal is then routed via an internal path around the outside of the XOR, to its output. The other duplications of the two signals are first combined via a logical AND operator (the two inputs of a confluent state), with the output of the AND then input to the component NOT operator. At output, the result of the NOT is then severally combined with the two duplicated signals to produce the output of the XOR. For the upstream XOR unit, the output is two first-stage (or partially) crossed signals.

At the micro-level, each NOT operator is composed of clocks, three types of gate, and internal paths. There are a total of fifteen clocks per NOT operator, five internal paths, and five of each kind of gate, with each gate being driven by one of the component clocks. Along each internal path, a sequence of the three kinds of gates, together with their driving clocks, is positioned. Each group of three gates implement one NOT operator. The five NOT operators function out-of-phase with each other, staggered to service every fifth bit of signal input. Hence, the clocks emit period five signals. The three gates, in order, select the bit to be inverted (the selector), invert the selected bit (the inverter), and collect the inverted bit (the collector) for transmission to output. The clocks of the selector and collector emit the signal <10000>, while the inverter emits the signal <11111>. The clock of the collector is out-of-phase with the clock of the selector, following by one state transition.

Given the descriptions above, we can compare the constructibility of the CC, MCO, and RTCO. The single internal path, the lack of interaction between component parts, and the lack of clock organs gives the CC the highest constructibility. Indeed, the presence of clock organs is the most important determinant of the unconstructibility of signal-crossing organs. The reason is simple: clocks are active organs. None of the component organs of the CC is active, while the MCO and RTCO each have several active component organs. We now look at the effect of active organs upon constructibility.

3 Configuration Construction

Constructibility is strongly impacted by configuration activity and the phasing of component clocks. All passive configurations are constructible. Unconstructible configurations express a signal, as suggested in figure 4. Yet, not all configurations that express a signal are unconstructible. In practice, the unconstructibility of a configuration has as much to do with the construction arm as with the configuration.

Configuration construction involves the sequential construction of individual cells, requiring a signal of between four and five bits, depending upon the desired end state of the constructed cell. For instance, confluent cell construction requires signal <1111> while construction of a right-pointing ordinary transmission cell requires signal <10000>. At least four ticks of the system clock are required for the construction of a single cell. Another time cost to construction is the motion of the construction arm. The

Fig. 4. An unconstructible configuration, shown without signaling, in both expected final form (right), and during construction (left), where the clock signal is the twelve-bits <111111111111>. Construction fails upon attempt to initialise the clock via the ordinary transmission path of the construction arm: interference from the clock alters the signaling of the construction arm. The construction arm is outlined with a dashed box

process of construction begins with the extension of the construction arm, followed by a sequence of pairings of cell constructions and construction arm retractions, one pair per cell constructed. Extension of the construction arm requires the separate construction of four cells, and the annihilation of two cells (a minimum unit cost of 18 system clock ticks), while retraction requires the separate construction of seven cells, and annihilation of nine cells (a minimum unit cost of 37 system clock ticks). Construction proper occurs during construction arm retraction, so construction of a single cell requires at least 41 system clock ticks.

The proximity of an active clock may impede cell construction. This occurs in two ways. Either the clock interferes with the signaling of the construction arm, or the clock provides alternative signaling to the constructed cell. In general, the signal emitted by a clock is not the same as the signal used to construct a cell. Similarly, the emitted signal will not likely correspond to the signal needed for construction arm extension and retraction. The far more likely condition is that the emitted signal will corrupt the signaling of the construction arm, or alter the state of a constructed cell. Interference with the construction arm is the reason that the configuration shown in figure 4 is unconstructible.

Even where an active configuration is constructible, the presence of multiple clocks presents a special difficulty. The construction arm can perform only one operation at a time, such as passing a signal to a clock. Consider the case in which two clocks are initialised, with one clock already circulating its signal. In order for the second clock to remain in proper relative phasing, the signal to the construction arm must be accurately timed. This requires knowledge of the phasing of the already operating clock, the length of the construction arm, and perhaps other quantities.

One means to construct a configuration having active organs is the use of a dedicated pulser. This allows the *a priori* computation of proper signaling for the construction arm, and so ensures proper clock phasing. For the RTCO, the configuration size of this pulser is some 10^5 cells, three orders of magnitude larger than the configuration size of the RTCO. To a good approximation, such a pulser would produce a signal of some 10^4 bits.[2] The size of this dedicated pulser is some ten percent of the size of a complete self-replicator, sans the external store [4]. The phasing of

[2] Though we have not sufficient room in this paper to demonstrate the result, it has been computed.

multiple clocks is not at all a trivial problem, the MCO and RTCO being good examples. Through redesign, both the RTCO and MCO can be made more constructible, which we now demonstrate for the MCO.

4 Autoinitialisation

Mukhopadhyay anticipated the difficulty of clock phasing in signal-crossing configuration construction, suggesting that means might exist to set the timing of clocks as a consequence of configuration function. The technique involves the sampling of signal input, and using the sample to trigger component clocks. We call this method autoinitialisation, and recognise the inherent implications of the technique for machine models of biological development, with particular respect to control of gene expression.

Configurations capable of autoinitialisation include subconfigurations (or AI organs) dedicated to the process. These AI organs sample signal inputs to the configuration, and generate control and clock signals. For many configurations, autoinitialisation is a one-time operation, occuring only for the first input signal. We now consider the case of an MCO configuration that supports autoinitialisation, and note one AI organ per internal path. Figure 5 shows the design of an MCO that includes

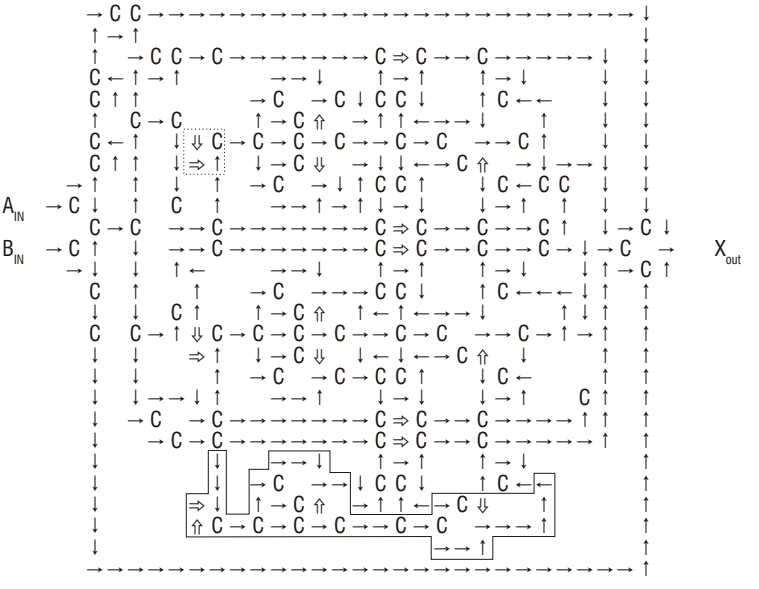


Fig. 5. An XOR unit of the MCO that supports autoinitialisation. Portals are indicated by two adjacent special transmission states, with one outlined with a dashed box. The autoinitialisation organ of one signal path is outlined with a solid line, at the bottom of the figure. The configuration shown is autoinitialised with the signal <11111>, applied synchronously at both inputs

AI organs. Each AI organ obtains signal input from the adjacent internal path of the MCO through a portal, generates properly phased signals for the component clocks of the corresponding selector, inverter, and collector gates, and generates a signal that closes the portal. Each portal consists of a confluent state inserted into the corresponding internal path. This confluent state duplicates the signal carried by the internal path, transmitting it both along the internal path, and to the AI organ, via adjacent ordinary transmission states that are pointed away from and perpendicular to the internal path. The portal is closed by changing this perpendicular ordinary transmission state into a confluent state. Closing the portal ensures that subsequent inputs to the configuration do not corrupt the signals of the clocks.

In addition to the increase in configuration size implied by the inclusion of AI organs, there is a cost to the time performance of the configuration. This cost comes in one of two forms. There is either a latency time cost or a propagation time cost associated with configuration function. In the case where the autoinitialisation is triggered by a dedicated signal, a design based upon latency is preferred, resulting in a one-time delay to the use of configuration function. If a dedicated signal is not available, then a design based upon propagation is the appropriate choice, with the delay borne for every use of the configuration. We term post construction changes to a configuration a reconfiguration. The MCO reconfigurations described are fairly simple to implement. Reconfigurations can be much more complex operations, involving dynamic change to a configuration, and they may be linked hierarchically.

One expects many layers of autoinitialisation to provide examples of machine-modeled biological development. For instance, the sequence of events in the cell cycle of bacteria consists of three basic phases: regeneration, regulation, and replication. The mechanisms of autoinitialisation suggest cellular automata replication can be extended into the processes of regeneration and regulation. Thus, with cascades of autoinitialisation giving rise to change in the function of a configuration, we can see opportunity for modeling replication in terms of regulation, instead of as a stand-alone (holistic) process. Further, if some operations in the cascade result in the expression of organs not present at replication, then regeneration becomes apparent. It is not so hard to envision a configuration that functions through such a sequence, thereby expressing a crude model of the cell development process.

5 Conclusions

As we have shown, solutions to the signal crossing problem in von Neumann cellular automata are hard to construct. We have developed the technique of autoinitialisation, and have demonstrated its application, yielding great improvement in configuration constructibility. For instance, the technique is easily applied to the task of post-construction initiation of clocks. Autoinitialisation can be extended to facilitate the run-time alteration of configurations. Systems of hierarchical autoinitialisation have strong implications for the modeling of the cell cycle. We expect that extension of the ideas presented here will yield computational models of biological developmental processes.

References

1. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana and London (1966)
2. Mukhopadhyay, A. Representation of Events in the von Neumann Cellular Model. J. of the ACM, Vol. 15, No. 4, October 1968, pp. 693-705
3. Burks, A. W. (ed.): Essays on Cellular Automata. University of Illinois Press, Urbana and London (1970)
4. Mange, D., Stauffer, A., Peparolo, L., Tempesti, G.: A Macroscopic View of Self-Replication. Proc. of the IEEE, Vol. 92, No. 12, December 2004, pp. 1929-1945

Evolutionary Discovery of Arbitrary Self-replicating Structures

Zhijian Pan and James Reggia

University of Maryland, Computer Science Dept. & UMIACS,
A. V. Williams Building, College Park, MD 20742, USA
{zpan, reggia}@cs.umd.edu

Abstract. In this paper we describe our recent use of genetic programming methods to automatically discover CA rule sets that produce self-replication of arbitrary given structures. Our initial results have produced larger, more rapidly replicating structures than past evolutionary models while requiring only a small fraction of the computational time needed in past similar studies. We conclude that genetic programming provides a very powerful tool for discovering novel CA models of self-replicating systems and possibly other complex systems.

1 Introduction

In the past studies of self-replicating CA structures, the rule sets governing cell state changes have generally been hand-crafted[1,2,3,4]. An alternate approach, inspired by the successful use of evolutionary computation methods to discover novel rule sets for other types of CA problems [5,6], used a genetic algorithm to evolve rules that would support self-replication [7]. This latter study showed that, given small but arbitrary initial configurations of non-quiescent cells ("seed structures") in a two-dimensional CA space, it is possible to automatically discover a set of rules that make the given structure replicate. However, some clear barriers clearly limited the effectiveness of this approach to discovering state-change rules for self-replication. First, to accommodate the use of a genetic algorithm, the rules governing state changes were linearly encoded, forming a large chromosome that led to enormous computational costs during the evolutionary process. In addition, as the size of the initial configuration increased, the yield (fraction of evolutionary runs that successfully discovered self-replication), decreased dramatically. As a result, it only proved possible to evolve rule sets for self-replicating structures having no more than 4 components, even with the use of a supercomputer, leading to some pessimism about the viability of evolutionary discovery of novel self-replicating structures.

In this paper, we revisit the issue of using evolutionary methods to discover new self-replicating structures and show that this earlier pessimism may be misplaced. We describe an innovative structure-encoding mechanism (S-tree) and a tree-like rule encoding mechanism (R-tree). As a result, genetic programming (rather than genetic algorithm) operators can be used. The resulting evolutionary system is qualitatively

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 404 – 411, 2005.
© Springer-Verlag Berlin Heidelberg 2005

more efficient and powerful than earlier methods, allowing the discovery of larger self-replicating structures with a standard computer rather than a supercomputer.

2 S-trees: General Structure-Encoding Representations

In the context of our work, an arbitrary structure can be viewed as a configuration of active cells in a CA space that satisfies two conditions. First, the active cells must be contiguous. Second, the configuration must be isolated from its environment. It follows that an arbitrary structure can be modeled as a connected, undirected graph, as we show in the following. The problem of structure encoding can then be converted to searching for a minimum spanning tree (MST) in order to most efficiently traverse the graph and encode its vertices (components).

Fig. 1 shows a simple structure in a 2D CA space, composed of 4 oriented components and satisfying the above two conditions. We convert the structure into a graph simply by adding an edge between each component and its 8 Moore neighbors, as shown in Fig. 2. The quiescent cells, shown empty in Fig. 1, are visualized with symbol "*" in Fig. 2. From this example we can see such a graph has the following properties: 1) it connects every component in the structure; 2) it also includes every quiescent cell immediately adjacent to the structure (which isolates the structure from its environment); and 3) no other cells are included in the graph. We name such a graph the *Moore graph*.

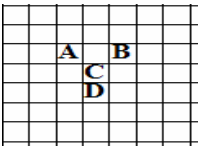


Fig. 1. The structure

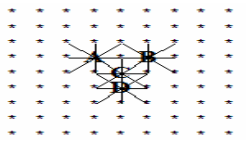


Fig. 2. The Moore graph

Fig. 3. The S-tree

Having the Moore graph for an arbitrary structure, we can then further convert the graph into a MST that we call the S-tree. Assigning a distance of 1 to every edge, we arbitrarily pick one component of the structure as the root, and perform a breadth-first-search of the Moore graph. The resultant tree is shown in Fig. 3. The essential idea is as follows. Starting from the root (A, in this example), explore all vertices of distance 1 (immediate Moore neighbors of the root itself); mark every vertex visited; then explore all vertices of distance 2; and so on, until all vertices are marked.

The S-tree therefore is essentially a sub-graph of the initial Moore graph. It has the following desirable properties as a structural encoding mechanism: 1) it is acyclic and unambiguous, since each node has a unique path to the root; 2) it is efficient, since each node appears on the tree precisely once, and takes the shortest path from the root; 3) it is universal, since it works for arbitrary Moore graphs and arbitrary CA spaces; 4) quiescent cells can only be leaf nodes; 5) active cells may have a maximum

of 8 child nodes, which can be another active cell or a quiescent cell (the root always has 8 child nodes); 6) it is based on MST algorithms, which have been well studied and run in near-linear time. Is the S-tree unique for a given structure? The MST algorithm only guarantees the vertexes of distance d to the root will be explored earlier than those of distance d+1. However, each Moore neighbor of a visited component lies the same distance from the root (such as B and D in Fig. 2), which may potentially be explored by the MST algorithm in any order and therefore generate different trees. This problem may be resolved by regulating the way each active cell explores its Moore neighbors, without loss of generality. For instance, let the exploration be always in a clock-wise order starting at a specific position (for instance, the left). As a result, we are guaranteed that a specific structure always yields the same S-tree. We say the resulting S-tree is in phase I, II, III, or IV, respectively, if the selected position is top, right, bottom, or left. The S-tree shown in Fig. 3 is in phase I. Fig. 4 shows the other phases. As clarified later, the concept of phase is important in encoding or detecting structures in rotated orientations.

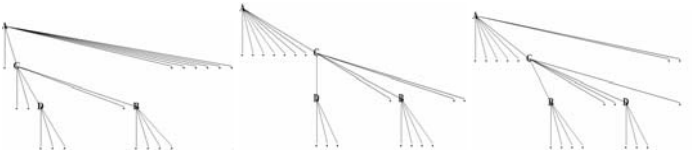


Fig. 4. S-tree at phase II, III, IV (from left to right)

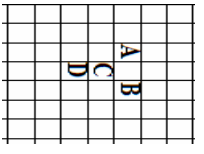


Fig. 5. Rotated Structure

We can easily convert an S-tree to a string encoding, simply by traversing the S-tree in a breadth-first order, and concatenating the state of each visited node to an initially empty string. The S-tree string encoding inherits the desirable properties of S-tree itself. It provides an unambiguous, efficient, and universal mechanism for representing an arbitrary structure, which enables an artificial evolution and rule learning system to be built and function without requiring the knowledge of any details of the involved structures a priori. Corresponding to the S-tree, there may be 4 different phases of S-tree encoding for a given structure. For each specific phase, the S-tree encoding is unique. Fig. 6 shows the S-tree encoding at each phase corresponding to the structure in Fig. 1.

S-tree encoding (Phase I) = " 1 0 0 0 9 0 0 0 0 5 0 0 13 0 0 0 0 0 0 0 0"
S-tree encoding (Phase II) = " 1 0 9 0 0 0 0 0 0 0 0 0 13 0 5 0 0 0 0 0 0 0"
S-tree encoding (Phase III)= " 1 0 0 0 0 0 0 0 913 0 5 0 0 0 0 0 0 0 0 0"
S-tree encoding (Phase IV)= " 1 0 0 0 0 0 9 0 0 5 0 0 13 0 0 0 0 0 0 0 0"

Fig. 6. The S-tree encoding at phases I, II, III, and IV

Note that the actual state index, rather than the symbol, of each component is used. This helps to distinguish the same component at different orientations. Also, to elimi-

nate any potential ambiguity, each state index takes two characters. Therefore, the spaces in the S-tree encoding are important.

In the CA space, a structure may be translated, rotated, and/or permuted during processing. The S-tree encoding can handle each of these conditions. First, since the S-tree encoding is independent of absolute position, it can be used to detect a structure arbitrarily translated. Second, the S-tree indicates that a string encoding at 4 different phases is equivalent to the structure rotated to 4 different orientations. Therefore, by detecting the way the S-tree phase has been shifted, the model can determine how the structure has been rotated. Further, if the structure's components have weak symmetry, the rotation of the structure will also cause the state of its individual components to be permuted. This can be handled by permuting each state by 90° every time the S-tree encoding shifts its phase. For instance, S-tree at phase II of the structure shown in Fig. 5 is identical to the S-tree at phase I of the structure shown in Fig. 1.

3 R-trees: General Rule Set Encoding

A CA rule determines the state of a cell at time t+1 based on the states of the cell and its adjacent neighbors at time t. The complete set of such rules, called the rule table, determines the state transition of each cell in the CA space. The previous study evolving rules for self-replicating CA structures adopted a linear encoding of the rules [7]. The essential idea is that the rule table took the form of a linear listing of the entire rule set. Each rule was encoded as a string CTRBLC', where each letter specifies respectively the current states of the Center, Top, Right, Bottom, and Left cells, and the next state C' of the Center cell. Let's denote the total number of states as N_s. The rule table will contain $(N_s)^5$ individual rules. The simple structure shown in Fig. 1 has 17 states, so a huge rule table of 17^5 = 1,419,857 rules is needed. This means that each cell has to make, in the worst case, 5 x 17^5 = 7,099,285 comparisons for a single state transition. Second, genetic operators have to manipulate individuals in such an enormous search space that computational barriers become prohibitive for the rule table to evolve effectively when the structure's complexity is moderately increased [7]. This section introduces R-tree encoding, which is much more efficient and effectively resolves the limitations of linear encoding.

An R-tree is essentially a rooted and ordered tree that encodes every rule needed to direct the state transition of a given structure, and only those rules. The root is a dummy node. Each node at level 1 represents the state of a cell at time t. Each node at level 2, 3, 4, and 5 respectively, represents the state of each von Neumann neighbor of the cell (without specifying which is top, left, bottom, and right). Each node at level 6 (the leaf node) represents the state of the cells at time t+1. An example R-tree is shown in Fig. 7, which has an equivalent rule table shown in Fig. 8. Rule 1 corresponds to the leftmost branch going to the 1st (leftmost) leaf, rule 2 corresponds to the 2nd leaf, etc.

The R-tree has the following properties: 1) it is a height balanced and parsimonious tree, since each branch has precisely a depth of 6; 2) the root and each node at level 1, 2, 3, and 4 may have maximum N_s child nodes, which are distinct and sorted by the state index; 3) each node at level 5 has precisely one child, which is a leaf; 4) it handles arbitrarily rotated cells with a single branch and therefore guarantees that there

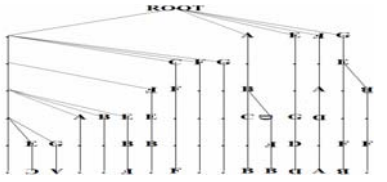


Fig. 7. An example R-tree

```
Rule 1:  0  0  0  0  0  -> 0
Rule 2:  0  0  0  0 17  -> 11
Rule 3:  0  0  0  0 25  -> 3
Rule 4:  0  0  0  1  0  -> 0
Rule 5:  0  0  0  0  5  -> 0
Rule 6:  0  0  0  0 17  5  -> 23
Rule 7:  0  0  0 23 17  5  -> 0
Rule 8:  0  0  9 21  0  0  -> 21
Rule 9:  0  0 21  0  0  0  -> 0
Rule 10: 0  0 25  0  9  0  -> 0
Rule 11: 1  1  0  5  9  0  -> 5
Rule 12: 1  0  0  5 16 23  -> 5
Rule 13: 17 0  0  0 25 13  -> 15
Rule 14: 23 0  1  0 15  0  -> 1
Rule 15: 25 17 0  9  0 21  -> 7
Rule 16: 25 17 7  0  0 21  -> 0
```

Fig. 8. The equivalent rule table

always exists at most one path that applies to any cell at any time, even after rotating and or permuting its orientation.

Due to the R-tree properties described above, the worst search cost for a single state transition is reduced to $5\ln(N_s)$ (5 nodes on each path to leaf, each has maximum N_s child nodes, ordered for quicksort search). Therefore, the ratio of the run cost between linear and R-tree encoding is: $5(N_s)^5/5\ln(N_s) = (N_s)^5/\ln(N_s)$. This means, for a simple structure shown in Fig. 1, R-tree encoding is $(17)^5/\ln(17) \approx 500{,}000$ times more efficient than linear encoding. The more complex a CA structure is, the better an R-tree encoding will outperform the linear encoding.

R-trees also allow efficient genetic operations that manipulate sub-trees. As with regular genetic programming, the R-tree crossover operator, for instance, swaps sub-trees between the parents to form two new R-trees. However, the challenge is to ensure that the crossover operator results in new trees that remain valid R-trees. If we simply pick an arbitrary edge E_1 from R-tree$_1$ and edge E_2 from R-tree$_2$, randomly, and then swap the sub-trees under E_1 and E_2, the resulting trees, for example, may no longer be height balanced. This problem can be resolved by restricting R-tree crossover to be homologous one-point crossover. The essential idea is as follows. After selecting the parent R-trees, traverse both trees (in a breadth first order) jointly in parallel. Compare the states of each visited node in the two different trees. If the states match, mark the edge above the node as a potential crossover point (PCP). As soon as a mismatch is seen, stop the traversal. Next, pick an edge from the ones marked as PCP's, with uniform probability, and swap the sub-trees under that edge between both parent R-trees.

R-tree crossover as defined above has clear advantages over linear-representation crossover. First, R-tree crossover is potentially equivalent to a large set of linear crossovers. Second, linear crossover randomly selects the crossover point and hence is not context preserving. R-tree crossover selects a crossover point only in the common upper part of the trees. This means that until a common upper structure emerges, R-tree crossover is effectively searching a much smaller space and therefore the algorithm quickly converges toward a common (and good) upper part of the tree, which cannot be modified again without the mutation operator. Search incrementally concentrates on a slightly lower part of the tree, until level after level the entire set of trees converges. The R-tree mutation operator simply picks an edge from the entire tree with uniform probability, and then eliminates the sub-tree below the edge. The R-tree encoding and genetic operators used allow CA rules to be constructed and evolved under a non-standard schema theorem similar to one proposed for genetic programming [8], even though R-trees do not represent conventional sequential programs.

4 Genetic Programming with S-Trees and R-Trees

Given an arbitrary structure for which a R-tree is sought to make the structure self-replicating, the seed is first encoded by an S-tree string, and then the R-tree is evolutionarily synthesized as follows:

```
Evolve_RTree (S, T, p_c, p_m)
    S: the R-tree population size
    T: the tournament selection size
    p_c: the fraction of S to be replaced by crossover at each generation
    p_m: the fraction of S to be replaced by mutation at each generation
 Initialization
    - Encode seed configuration as an S-tree string
    - Initialize Current_Population, with R-trees each with one branch "ROOT . . . . . "
    - Max_Time = 1, Terminate = false
 WHILE Terminate == false DO
     FOR each R-tree in Current_Population DO
        Each CA cell advances time from 0 to Max_Time, directed by current R-tree
        IF missing rule condition THEN
          allow the R-tree to self-expand, with the leaf state randomly selected
        ENDIF
        Compute the fitness of the R-tree based on the S-tree encoding
        Prune inactive branches in the R-tree
     ENDFOR
     IF terminate condition THEN
           - Terminate = true
     ELSE IF fitness no longer improves THEN
           - Max_Time = Max_Time + 1
     ENDIF
     FOR RTree_Pair from 1 to S/2 DO
           - Randomly pick two parent R-trees using tournament selection.
           - Generate a random number p in (0,1)
           IF p_c > p THEN
                -Perform crossover and store offspring R-trees in Temporary_Population
           ELSE
                -Directly store the parents in Temporary_Population
          ENDIF
     ENDFOR
     FOR each R-tree in Temporary_Population DO
           - Generate a random number p in (0,1)
           IF p_m > p THEN
                - Mutate the R-tree
           ENDIF
     ENDFOR
     SET Current_Population = Temporary_Population
 ENDWHILE
 RETURN the R-tree with highest fitness
```

In the algorithm depicted above, "missing rule condition" means no path/leaf in the R-tree applies to change that cell's state even after rotating and permuting its von Neumann neighbors, "terminate condition" means finding a set of rules capable of constructing the replicated structures or reaching a pre-specified maximum number of iterations, and "fitness no longer improves" means the best fitness at each generation is not further increased, after a configurable number, say 300, of continuous GP generations. Therefore, only gradually do the number of CA iterations increase over time as fitness im-

proves; this was an important factor in controlling the R-tree size and increasing algorithm efficiency. Typically we used S = 100, T = 3, $p_c = 0.85$, and $p_m = 0.15$.

The fitness of an R-tree is evaluated in terms of how well the states it produces "match" the structural information encoded in the S-tree. More specifically, the following fitness functions are defined: 1) the matched density measure f_d evaluates how many components appearing in the S-tree encoding are detected; 2) the matched neighbors measure f_n evaluates how many Moore neighbors of the components found above also match the neighborhood encoded by the S-tree encoding; 3) the matched component measure f_c evaluates how many components found above have their Moore neighbors perfectly matching the S-tree encoding; and 4) the matched structure measure f_s evaluates the number of root components which perfectly match the entire S-tree encoding. The overall R-tree fitness function is then defined as: $f = w_d * f_d, + w_n * f_n, + w_c * f_c, + w_s * f_s$. Typical weights that we used were: $w_d = 0.1$, $w_n = 0.1$, $w_c = 0.4$, $w_s = 0.4$. In the early generations of the evolutionary process described above, f_d encourages the same components to appear in the cellular space as in S-tree encoding (other measures will likely be near 0 at this phase). Early or late, f_n comes into play and rewards the R-trees which tend to organize components to form a neighborhood that is the same as in the Moore graph. Naturally, sometimes components will appear with Moore neighbors perfectly matching the S-tree, and so f_c will cause a significant jump in the overall fitness. Eventually, the perfectly matched components may form replicates of the original structures, which will be strongly rewarded by f_s.

In sum, the R-tree encoding is evolutionarily, adaptively, incrementally, and parsimoniously self constructed from the S-tree encoding, through genetic programming. As a result, replicates of an arbitrary seed structure can be synthesized.

```
S-tree encoding(I)  =" 1 0 0 513 9 0 0 0 0 0172521 0 0 0 0 0 0 0 0 0"
S-tree encoding(II) =" 1 513 9 0 0 0 0 0 017 02521 0 0 0 0 0 0 0 0 0"
S-tree encoding(III)=" 1 9 0 0 0 0 0 0 513 0 021 0 01725 0 0 0 0 0 0 0"
S-tree encoding(IV) =" 1 0 0 0 0 0 513 9 0 0 0172521 0 0 0 0 0 0 0 0 0"
```

Fig. 9. The S-tree encoding

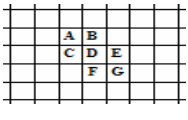


Fig. 10. The seed

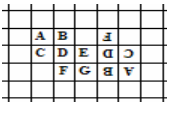


Fig. 11. At $t = 1$

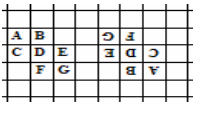


Fig. 12. At $t = 2$

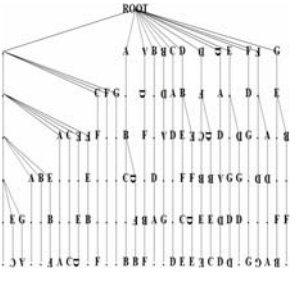


Fig. 13. The R-tree

5 Experimental Results

The model described above was tested in a number of experiments. We achieved success with structures of arbitrary shape and varying numbers of components. The largest seed structure for which it was previously possible to evolve rules with over a week's computation on a super-computer had 4 components [7]. Figure 10 shows one of the seed structures, consisting of 7 oriented components, for which our approach found a rule set that allowed the structure to self-replicate. The R-tree (Fig. 13) was evolved from the S-tree encoding (Fig. 9) after about 20 hours of computation on an

IBM ThinkPad T21 laptop. With the resultant R-tree, at time t=1 (Fig. 11), the structure starts splitting (the original translates to the left while a rotated replica is being born to the right). At time t=2 (Fig. 12), the splitting completes and the original and replica structures become isolated. Thus, the seed structure has replicated after only 2 time steps, a remarkably fast replication time that has never been reported before. As time continues, more replicas appear, with debris remaining between replicas (not illustrated). These experiments suggest that our model is much more efficient than previous genetic algorithm models [7].

6 Conclusions and Future Work

In this article, we introduced an S-tree—R-tree—structure synthesis model coupled with genetic programming methods. Our experimental results so far indicate that such a model is indeed capable of evolving rule sets that make arbitrary structures of limited size self replicate, as well as efficient computation. There is much room for further study and additional experiments. For instance, one motivation for the S-tree encoding is that it should eventually allow both structure and rules to evolve concurrently and cooperatively. The S-tree and R-tree encoding might also be used to evolve rule sets replicating extra structures in addition to the seed itself, or structures with higher complexity than the seed, etc. Acknowledgements: JR's work on this project is supported by NSF award IIS-0325098.

References

1. J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Illinois, 1966. Edited and completed by A. W. Burks.
2. Sipper, M. (1998). Fifty years of research on Self-Reproduction: An overview. *Artificial Life*, 4, 237-257.
3. Langton C, Self-Reproduction in Cellular Automata, *Physica D*, 10, pp. 135-144, 1984.
4. Reggia J, Armentrout S, Chou H & Peng Y, Simple Systems That Exhibit Self-Directed Replication, *Science*, 259, 1282-1288, 1993.
5. Andre D, Bennett F, & Koza J. Discovery by Genetic Programming of a Cellular Automata Rule ..., Proc. First Ann. Conf. on Genetic Programming, MIT Press, 1996, 3-11.
6. Richards F, Meyer T & Packard N. Extracting cellular automaton rules directly from experimental data, Physica D, 45, 1990, 189-202.
7. Lohn J & Reggia J. Automated discovery of self-replicating structures in cellular automata.' IEEE Trans. Evol. Comp., 1,1997, 165-178.
8. Poli R and Langdon W, Schema theory for genetic programming with one-point crossover and point mutation, Evolutionary Computation, 6, 231-252, 1998.

Modelling Ant Brood Tending Behavior with Cellular Automata

Daniel Merkle, Martin Middendorf, and Alexander Scheidler

Department of Computer Science, University of Leipzig,
Augustusplatz 10-11, D-04109 Leipzig, Germany
{merkle, middendorf, scheidler}@informatik.uni-leipzig.de

Abstract. The brood sorting behavior of ants like Leptothorax unifasciatus leads to patterns, where brood items are sorted in concentric circles around the nest center. The underlying mechanisms are not fully understood so far and brood tending simulations can help to explain the occurrence of these patterns. We reexamine an existing cellular automata based model for ant brood tending. This model is then modified and extended by including a carbon dioxide distribution in the nest, that influences the ants movement behavior. Furthermore, the ants can deliver food to the brood in our model. Results of simulation runs are presented that can help to explain brood patterns that have been observed in natural ant colonies.

1 Introduction

Brood sorting and brood tending behavior in ants has inspired several novel methods in computer science like clustering algorithms or multi robot systems (e.g., [2]). The underlying mechanisms which lead to such complex behavior are still under investigation. In [3] the pattern formation within Leptothorax unifasciatus ant colonies was investigated in artificial nests. The youngest brood items (eggs and microlarvae) are placed in the center, successively larger larvae are arranged in concentric rings around the center. However, the largest and oldest brood (pupae and prepupae) is placed in an intermediate area between the peripheral larvae and the larvae of medium size. One suggestion why this happens is that these patterns help to organize the brood care [3]. The most valuable brood has to be fed first and is therefore placed at the outside, the pupae and prepupae may be placed in intermediate positions as they do not need food but only grooming.

In the inspiring paper [7] a stochastic lattice gas model of ant brood tending was formulated. The authors suggested, that a possible reason for the central egg location is that brood care is more evenly distributed in that area. We reexamine this model, make modifications to avoid unwanted artifacts, and extend it by taking a CO_2 distribution in the nest into account that influences the ants movement behavior. Moroever, we simulate the brood feeding behaviour of the ants. The paper is structured as follows. Section 2 reviews the chemical

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 412–419, 2005.
© Springer-Verlag Berlin Heidelberg 2005

and biological background which is needed in the models. The cellular automata transition functions used in [7] and our modifications are presented in Section 3. The model and the results from [7] are reexamined in Section 4. Experimental results with our model are discussed in Section 5 and a conclusion is given in Section 6.

2 Biological Background

Several studies have shown the influence of carbon dioxide (CO_2) concentration on the behavior of social insects (e.g., [6]). In [1] a diffusion model was introduced and spatial patterns of CO_2 concentration were explored analytically and numerically. A special role plays the concentration gradient, as it is qualitatively independent of fluctuations of the absolute concentration. Ants are likely to have the ability to detect this gradient with their antenna and are able to infer regions of CO_2 sources (e.g., nest center) and CO_2 sinks (e.g., the nest entrance or periphery). In [8] the ability to detect the direction of the colony center was used to model worker sorting in ant colonies. This approach was based on a model in [4] for bacteria moving along a chemical gradient.

Besides the knowledge of the nest center the ants in our model will also carry food. Ants carrying food tend to move towards the nest center, ants without food tend to move to the periphery of the nest. Each ant will drop the food with a certain probability.

3 Transition Functions and Measures for Brood Care

This section describes the transition functions of the cellular automata models which are investigated in this paper. For all models the two dimensional Moore neighborhood $N = \{(-1,1),\ldots,(1,1)\}$ and a space $R = \{1,\ldots,L\} \times \{1,\ldots,L\}$ is used. At any lattice site there can only be 0 or 1 ant. In all models we proceed similar to the cellular automata model in [7]: Within one time step all ants move asynchronously and in random order. If an ant wants to move onto a lattice site, that is occupied by another ant, it will not move. The driving rate p is the probability that ants occur on the boundary square, an ant which moves onto a boundary lattice is deemed to have left the brood, and is removed. Note, that the following description of the ants movement behavior could easily be transformed into a stochastic transition function for a cellular automata.

3.1 Transition Functions for Modelling Ant Movement Behavior

Deterministic movement: The deterministic behavior of the ants as described in [7] is based on the transition function used in [5]. An ant tries to move in a direction which takes it as far as possible from the other ants of the four neighboured cells (north, south, west and east). Formally, let n be the preferred direction of motion for an ant, which is determined as $n = [F_x/F]i + [F_y/F]j$ where $F_x = a(r-i,t) - a(r+i,t)$, $F_y = a(r-j,t) - a(r+j,t)$, and $F = \sqrt{F_x^2 + F_y^2}$,

i and j are the standard basis vectors in two dimensional space, operator $[x]$ means the nearest integer to x, $a(r,t)$ is the number of ants located in cell r at time t. In [7] the tendency of ants to move in direction to the nest center is modelled by placing a ring of stationary ants around the actual space R, thus an ant which occurs on a boundary square of R will be forced to move away from the boundary (a similar strategy was used in [5]).

Probabilistic movement behavior: To incorporate stochastic elements to the ants behavior the following strategy was suggested in [7]. Let n be the preferred direction according to the deterministic behavior of the ant, and $d(n^*) = |n^* - n|$ be a function that quantifies the deviation from direction $n \in N$ to a direction $n^* \in N$. The probability $P^{n,\beta}$ of moving to n^* is calculated as

$$P^{n,\beta}(n^*) = \frac{e^{-\beta d(n^*)}}{\sum_{n^* \in N} e^{-\beta d(n^*)}} \tag{1}$$

where β is a parameter that determines the degree of randomness of the movement direction of an ant. If $\beta = 0$ all directions are equally likely, if $\beta \to \infty$ the probability distribution in Equation 1 approaches a delta function peaked at $n^* = n$, i.e., ants behave according to the deterministic movement.

Brood Care Measure: It is argued in [7] that brood tending is of less quality, if brood care is unevenly distributed. Therefore, let $\tau(r,t)$ be the amount of brood care (i.e., how often an ant was located at cell r within the first t steps), let $\overline{\tau}(t) = 1/L^2 \sum_{r \in R} \tau(r,t)$ be the mean tending time per brood item, and $\sigma(t) = \left[1/L^2 \sum_{r \in R} (\tau(r,t) - \overline{\tau}(t))^2\right]^{1/2}$ be the standard deviation of brood tending times. Then the relative fluctuation amplitude of brood tending times, $\sigma^*(t) = \sigma(t)/\overline{\tau}(t)$ is a dimensionless measure of fluctuations in the amount of brood care. A value of $\sigma^*(t) < 0.1$ is considered as efficient brood tending.

3.2 Transition Function for the Extended Model

In this subsection we will present our extended model for ant brood tending which incorporates a CO_2 distribution within the nest and the ability of ants to deliver food. Two new parameters are σ for the standard deviation of the two dimensional normal distribution, that is used for modelling the CO_2 pattern within the nest, and parameter f which is the probability that a food carrying ant drops it.

Strict gradient movement: For modelling the movement of ants towards or away from the nest center, we determine CO_2 levels in the nest according to a two dimensional normal distribution. The CO_2 level in cell $r = (x,y)$ is

$$g_r = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{1}{2}\left(\left(\frac{x-\mathbf{c}}{\sigma L}\right)^2 + \left(\frac{y-\mathbf{c}}{\sigma L}\right)^2\right)}$$

where $c = (\mathbf{c}, \mathbf{c})$ with $\mathbf{c} = \frac{L+1}{2}$ is the center of the nest. An ant uses the gradient of these CO_2 levels to determine its movement. An ant that carries food is moving

towards the center of the nest, an ant without food is moving away from the center. Therefore, we use $I = \{(1,0),(0,1),(1,1),(-1,1)\}$ for determining the pheromone gradient sum $G_r = \sum_{i \in I} |g_{r+i} - g_{r-i}|$, and let $G^{\max} = \max_{r \in R} G_r$ be the maximal gradient sum in the nest, which will be used for normalization of the probabilistic behavior. Formally, the probability that an ant carrying food located at cell r will move in direction $n \in N$ is determined as follows:

$$P_r^g(n) = \begin{cases} \frac{|g_{r+n} - g_{r-n}|}{G^{\max}} & \text{if } \; n \neq (0,0) \;\text{ and }\; g_{r+n} > g_{r-n} \\ 1 - \sum_{m \in N \setminus (0,0)} P_r^g(m) & \text{if } n = (0,0) \\ 0 & \text{otherwise} \end{cases}$$

For an ant without food the same formula is used, but in the first case $g_{r+n} > g_{r-n}$ has to be exchanged by $g_{r+n} < g_{r-n}$. Note, that the probability to move is small (resp. large) in areas where the sum of gradients of the CO_2 level is small (resp. large).

Probabilistic movement behavior: To incorporate more randomness in the ants behavior which move according to the strict gradient movement, we used two methods. The first method is according to [7]. In contrast to the incorporation of randomness as given in Equation 1, there is no preferred direction n of an ant (which is used to calculate the probabilities $P^{n,\beta}$). Instead a probability vector P_r^g is used, that determines the probabilities that an ant located at cell r moves to a certain neighbor cell when strict gradient movement behavior is used. $P^{n_1,\beta}(n_2)$ is the probability that an ant with the preferred movement direction n_1 moves in direction n_2 (see Equation 1). Then

$$P_r(n) = \sum_{\substack{(n_1,n_2) \\ n_1+n_2=n}} P_r^g(n_1) \cdot P^{n_1,\beta}(n_2) \tag{2}$$

determines the probability that an ant located at cell s moves to neighbor n. Similar to Equation 1 each neighbor is equally likely for $\beta = 0$, and for $\beta \to \infty$ we have $P_r(n) = P_r^g(n)$, i.e., the ants behave according to the strict gradient movement.

The second method to incorporate more randomness simply uses a linear combination of the strict gradient movement behavior and a pure random behavior, i.e.

$$P_r(n) = (1 - \lambda) P_r^g(n) + \lambda(1/|N|) \tag{3}$$

where λ determines the degree of randomness ($\lambda = 0$ leads to a strict gradient movement behavior, and for $\lambda = 1$ a movement to any neighbor is equally likely).

Brood Care Measure: As suggested in [7] we count the number of brood tending times $\tau(r,t)$ of brood located at cell r within t steps. As in this model ants carry food, we will also measure the number $\zeta(r,t)$ of feeding times, i.e. how often food is dropped in cell r within t steps. For our investigations we will also use the mean tending time per brood member in certain areas around the center $c = (\mathbf{c},\mathbf{c})$ of the nest. Therefore, we measure $\overline{\tau_R}(t,k) = 1/|R_k| \sum_{r \in R_k} \tau(r,t)$

with $R_k = \{r \in R : k-1 \leq ||r-c|| \leq k\}, k \geq 1$. If for $||r-c||$ the $L2$ norm is used, this function is denoted by $\overline{\tau}_{circ}(t,k)$, if the L-infinity norm is used, it is denoted by $\overline{\tau}_{square}(t,k)$. We proceed similar for mean feeding times $\overline{\zeta}(t,k)$.

4 Reexamination of a Brood Tending Model

In this section we reexamine the results presented [7]. Brood care σ^* was measured on a field of size $L = 40$ for different degrees of randomness ($\beta \in \{0, 0.1, 0.5, 1, 3, 5, \infty\}$) and different driving rates $p \in \{0.05, 0.1, 0.2, 0.5, 1\}$. The brood care intensity values for the whole space and for the center of the brood (the central 12×12 lattice) are given in Figure 1.

In our simulation we basically obtained the same results as presented in [7], but we can not agree with their interpretation why the brood care is worse at the periphery. We consider this mainly as an artifact of the model. To show this

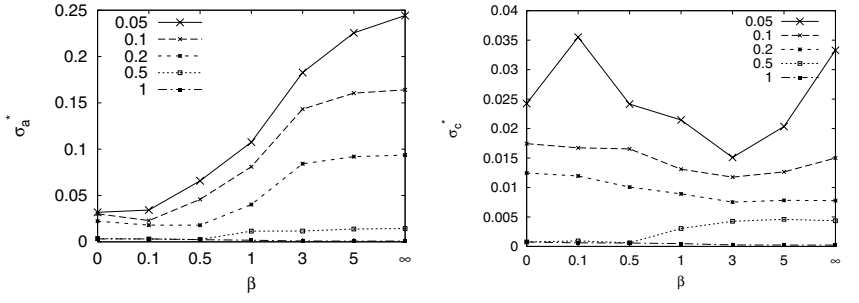


Fig. 1. Asymptotic relative fluctuation σ^* for different values of β and p (different curves) in the whole nest of size 40×40 (left) and in the center square of size 12×12

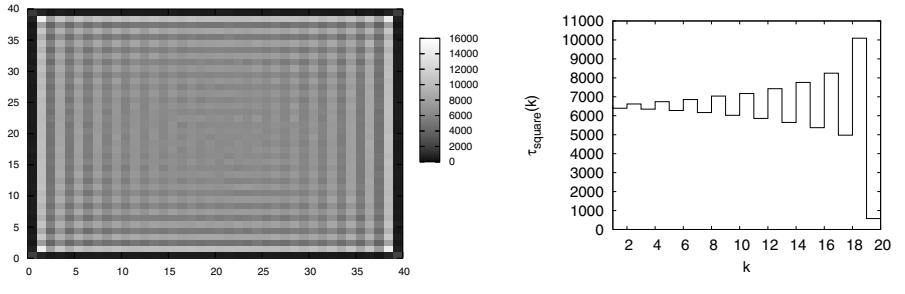


Fig. 2. Brood tending values τ; dark (resp. light) colors correspond to small (resp. large) values (left); mean brood tending $\tau_{square}(t,k), 1 \leq k \leq 20$ (right); parameters were $t = 20000$, $p = 0.05$, and $\beta = \infty$

we measured the brood care in every single cell (see in Figure 2). Suppose a β value is used, that combines a random behavior with the deterministic behavior (comp. Equation 1). As in the model of [7] there is a ring of stationary ants on the boundary. The influence of the deterministic behavior of the ants will lead to the effect, that ants which occur on the boundary tend to move towards the center. Therefore, the brood care in the ring of cells that abut on the boundary will be bad. The brood care in the next ring of cells towards the center will be better, this effect continues until the random influence gets too large, and brood care is more evenly distributed. The brood care after 20000 steps is shown for $\beta = \infty$ in Figure 2, the brood care values $\overline{\tau}_{square}$ are also given. Obviously the uneven brood care will lead to smaller σ^* values in the center of the nest.

5 Results for the Extended Model

In Figure 3 the tending behavior in the extended model is shown for $p = 0.05$, $f = 0.03$, $\lambda = 0.5$ and $\sigma = 0.5$. This result is particularly interesting, as the used parameter for the normal distribution of CO_2 in the nest leads approximately to the equilibrium distribution as given in [1] and can therefore be considered as a realistic distribution. In Figure 3 three different regions of brood tending can be distinguished. This helps to explain why the brood is organized in concentric rings around the nest center. To examine the brood tending (resp. brood feeding) behavior more exactly the average intensity of tending (resp. feeding) in concentric circles and squares around the center was measured (see Figure 4). Again, the different regions can be observed.

Brood tending results for different driving rates $p \in \{0.01, 0.03, 0.1\}$ are shown in Figure 5. The existence of different brood tending areas is obviously quite robust with respect to the driving rate. The influence of the degree of randomness in the ants behavior is much stronger, as can be seen in Figure 6. While for $\lambda = 0.9$ (nearly random movements of the ants) the smallest brood tending values are in the nest center, this is very different for a value of $\lambda = 0.7$, where three different tending areas appear. For even smaller values of λ the tending in the nest center becomes extremely large and does not model realistic ant tending behavior ($\lambda = 0.2$ in our simulation).

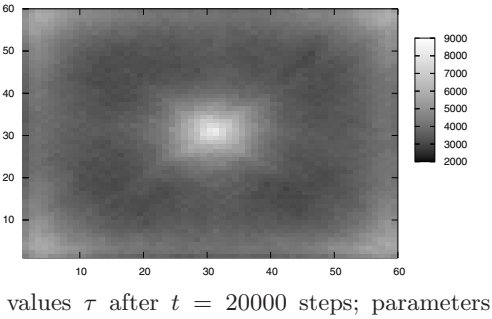


Fig. 3. Brood tending values τ after $t = 20000$ steps; parameters were $p = 0.05$, $\sigma = 0.5$, $\lambda = 0.5$, and $f = 0.03$

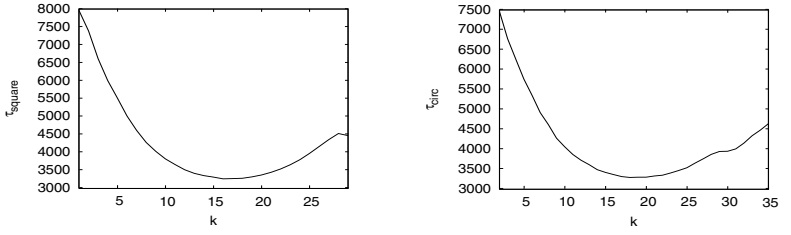


Fig. 4. Mean brood tending values $\overline{\tau}_{square}(t,k)$ (left) and $\overline{\tau}_{circ}(t,k)$ (right) for test run shown in Figure 3

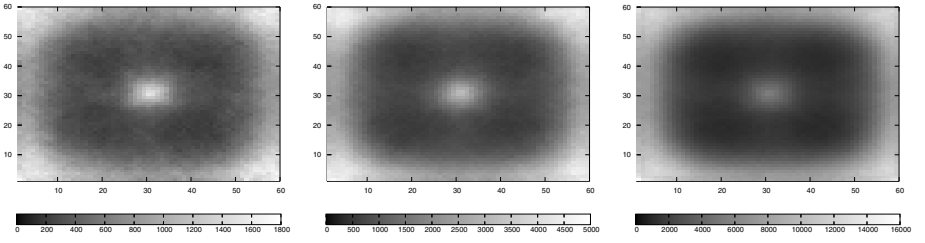


Fig. 5. Brood tending values τ for different driving rates $p = 0.01$ (left) $p = 0.03$ (middle), and $p = 0.1$ (right) after $t = 10000$ steps; parameters were $\sigma = 0.5$, $\lambda = 0.7$, and $f = 0.02$

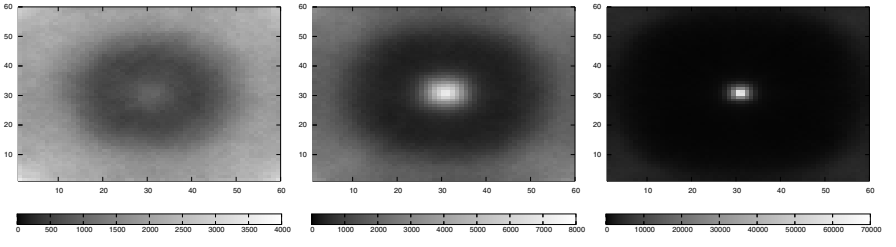


Fig. 6. Brood tending values τ for different degrees of randomness after 10000 steps; parameters were $\lambda = 0.9$ (left), $\lambda = 0.7$ (middle), $\lambda = 0.2$ (right), $p = 0.05$, $f = 0.008$, $\sigma = 0.25$

Due to space limitation we can only present the following results. Hence, results for other nest sizes or for the second strategy to combine strict gradient movement with randomness have to be omitted. Similar to [1] we investigated the brood tending behavior for situations with a different distribution of CO_2. Such distributions can occur when, e.g., only three sides of the nest are open, and the other side is not a CO_2 sink. This leads to a different equilibrium distribution of CO_2 and influences the behavior of the ants. Figure 7 shows the brood feeding and tending behavior for this case (up, left, and bottom) and a CO_2 distribution, where the largest concentration is not located in the center of the nest, but in

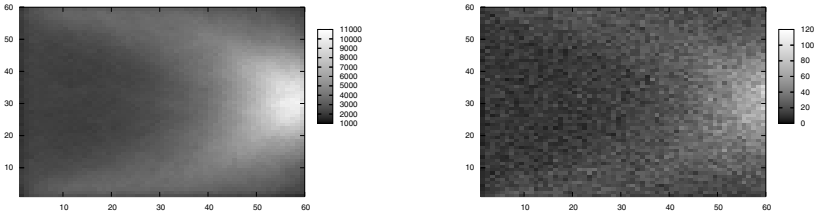


Fig. 7. Brood tending τ (left) and feeding ζ (right) values for three open nest sides (up, left, and bottom) and a CO_2 distribution, where the largest concentration is located in the middle of the right border; parameters were $p = 0.05$, $\lambda = 0.5$, $f = 0.02$, $\sigma = 1$, and $t = 10000$

the middle of the right border. It would be interesting to compare this results with experiments for such a situation with real ants.

6 Conclusion

In this paper a cellular automata model for ant brood tending behavior that uses a combination of deterministic and stochastic behavior was reexamined. We have shown that uneven brood care is an artifact of the model. A modification of this model has been introduced where uneven brood tending patterns occurs similar to those observed in real ant colonies. The model has also been extended to include the carbon dioxide distribution that influences the ants movement behavior and to model food delivering by the ants. Our results may help to explain the existence of pattern formation in real ant nests.

References

1. M.D. Cox and G.B. Blanchard. Gaseous templates in ant nests. *Journal of Theoretic Biology*, 204:223–238, 2000.
2. L. Deneubourg, S. Goss, N. Franks, A.B. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting: Robot-like ants and ant-like robots. In *Proc. of the 1st Int. Conf on Simulation of Adaptive Behavior*, pages 356–363, 1991.
3. N.R. Franks and A.B. Sendova-Franks. Brood sorting by ants: distributing the workload over the work-surface. *Behavioral Ecology and Sociobiology*, 30:109–123, 1992.
4. D. Grünbaum. Translating stochastic density-dependent individual behavior to a continuum model of animal swarming. *J. of Mathematical Biology*, 33:139–161, 1994.
5. H.J. Jensen. Lattice gas model of $1/f$ noise. *Phys. Rev. Lett.*, 64:3103–3106, 1990.
6. G. Nicolas and D. Sillans. Immediate and latent effects of carbon dioxide on insects. *Annual Review of Entomology*, 34:97–116, 1989.
7. D.V. O'Toole, P.A. Robinson, and M.R. Myerscough. Self-organized criticality in ant brood tending. *Journal of Theoretical Biology*, 221:1–14, 2003.
8. A.B. Sendova-Franks and J.V. Lent. Random walk models of worker sorting in ant colonies. *Journal of Theoretical Biology*, 217:255–274, 2002.

A Realistic Cellular Automata Model to Simulate Traffic Flow at Urban Roundabouts

Ruili Wang and Mingzhe Liu

Institute of Information Sciences and Technology, Massey University,
Private Bag 11222, Palmerston North 5301, New Zealand
{r.wang, m.z.liu}@massey.ac.nz

Abstract. In this paper a realistic cellular automata model is proposed to simulate traffic flow at single-lane roundabouts. The proposed model is built on fine grid Cellular Automata (CA), so it is able to simulate actual traffic flow more realistically. Several important novel features are employed in our model. Firstly, 1.5-second rule is used for the headway (=distance /speed) in car-following process. Secondly, vehicles movement on urban streets are simulated based on the assumption of speed changes following a Gaussian (normal) distribution and is calibrated with the field data. Thirdly, driver behavior is modeled by using a truncated Gaussian distribution. Numerical results show that our method is feasible and valid.

1 Introduction

Roundabouts, regarded as complex subsystems, are important components of complex urban networks. The most important control rule in roundabouts is yield-at-entry [1], i.e. vehicles from the secondary roads give way to the vehicles on the circulatory road. Both empirical and theoretical methods [1] have been proposed to measure roundabouts performance such as capacity, delay, queue length, etc. With regard to these methods, the gap-acceptance criteria such as in [2, 3] are commonly used. Gap-acceptance models are, however, unrealistic in general assuming that drivers are consistent and homogenous [4]. A consistent driver would be expected to behave in the same way in all similar situations, while in a homogenous population, all drivers have the same critical gap (the minimum time interval between two major-stream vehicles required by one minor-stream vehicle to pass through) and are expected to behave uniformly. The limitations of gap-acceptance models have been analyzed and detailed in literature [5]. Thus, in this paper we focus on using a Cellular Automata (CA) model to simulate traffic flow at an urban roundabout.

The employment of CA modeling traffic flow at roundabouts has attracted attention in the last few years [4-9], due to its dynamical and discrete characteristics [10] and its connection with stochasticity [11]. For a roundabout, vehicle maneuvers may include driving on the roads and on the roundabout.

Vehicles moving on the roads can be seen as driving on a straight urban road. Many models, such as in Ref. [7, 12, 13] have been developed to deal with driving on urban networks. To our knowledge, previous models normally implicitly assume that

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 420 – 427, 2005.
© Springer-Verlag Berlin Heidelberg 2005

the headways (=distance/speed) are 1 second, that is, 2-second rule is not considered in those models. Theoretically, it should be observed by all drivers, although the headways that drivers use are shorter than 2 seconds [14] and normally longer than 1 second in the real world. In our research we have recorded 10 hours of traffic data between 16 August 2004 and 27 August 2004. The average car-following headway of 1.5 seconds has been observed in local urban networks and this 1.5-second rule has been built into our model.

Wang and Ruskin [5] proposed a Minimal Acceptable sPace (MAP) method to simulate interactions between drivers at single-lane roundabouts. The method is able to simulate heterogeneous driver behavior and inconsistent driver behavior. In their model, driver behavior is randomly classified into four categories: conservative, rational, urgent and reckless, and each group has its own MAP. Meanwhile, inconsistent driver behavior is simulated by reassignment of categories with given probabilities at each time step. Although the assumption to categorize driver behavior into four groups is coarse, this approach, as far as we know, is the first model to reveal the impact of driver behavior on traffic flow at roundabouts.

Literature [7] proposed a stochastic CA interaction model. With this model, a waiting vehicle can enter the roundabout only if there are no vehicles on the roundabout in its left side quadrant. Obviously, this assumption is questionable. Each time step in the model is equivalent to 2 seconds. Clearly, the model is not able to describe traffic flow in details, such as acceleration or deceleration. A time step in micro-simulation is recommended between 0.1 and 1 second [2].

A simpler entry rule is also presented in [6], that is, if the cell located in front of the entrance is not occupied by the vehicle, a waiting vehicle is randomly generated and the cell is occupied. However, the yield-at-entry rule is not abided by and the speed of the following one on the circulatory lane is not considered, therefore, their model is unlikely to be a safety model.

We proposed a Normal Acceptable Space (NAS) model in this paper to describe heterogeneous driver behavior under normal conditions. The value of NAS is the number of required cells on a circulatory lane for a vehicle from a secondary road to enter the roundabout. The deviation of the NAS is used to model inconsistent driver behavior.

This paper is organized as follows. In Section 2, several important novel features are employed. Firstly, the average headway of 1.5 seconds is built into our model. Secondly, driver behavior and vehicle movement are modeled using a (truncated) Gaussian distribution. In this way, we present interaction rules at roundabouts according to the left-side driving such as in UK, Australia, and New Zealand. In Section 3, vehicle movement on urban roads is calibrated by field data and interaction models are also calibrated using field data provided in [19]. Furthermore, a comparison with other models is given and the result shows that our model is approximately consistent with other models. The conclusion is given in Section 4.

2 Model Formulations

In this paper a shorter length of cells is used in our model. In other words, the finer discretization of cells in our CA model is used comparing with previous models. The

length of each cell is equal to 1 m in a real road, which provides a better resolution modeling actual traffic flow than other models. A unit of speed is therefore equal to 3.6 km/h and each time step is 1 second. Since 1 unit of acceleration is 1 m/s^2, this also corresponds to a 'comfortable acceleration' [15].

In urban networks, a lower speed should be considered due to speed constraints. Normally, the legal limit of speed in urban networks is 50 km/h, however some people will drive at speeds about 58 km/h, which is just below the limit (61 km/h) of being apprehended. Therefore, in our model, we assume the maximum speed of each vehicle is in the range of 50.4 km/h – 57.6km/h. The speed corresponds to the number of cells, which a vehicle can move forward to in 1 second. The number of cells is 14-16 cells.

Different vehicle types have different numbers of cells in lengths. Following are average values based on 10-hour recording data sets at morning peak hour and these are adopted in this paper.

Table 1. Vehicle components and required cells

Vehicle Types	Occupied Cells	Percentage (%)
Motorcycles (M)	3	2
Personal Vehicles (P)	5	78
Vans and minibuses (V)	7	11
Buses (B)	10	6
Other large vehicles (O)	13	3

2.1 Modeling Driver Behavior Under Gaussian Distribution

As mentioned above, driver behavior is inconsistent, namely, even under similar conditions a driver may behave differently with time. So a driver can accept a space which value is shorter than the NAS due to long waiting time or other urgent conditions. Sometimes, a driver needs a space, which value is larger than the NAS due to bad weather, night visibility or other factors.

Let x_{min} represent the number of minimum acceptable cells and x_{max} stand for the number of maximum acceptable cells for a driver to interact with other drivers. If $x > x_{max}$, a vehicle surely enters the roundabout without delay, but there is no interaction with other drivers. The values less than x_{min} are rejected due to safety factors and the values larger than x_{max} are not included in consideration as no interaction is needed (free flow). Therefore, the model can be viewed as a *truncated* Gaussian distribution [16], where the left and right parts have been cut off. Mathematically, the *truncated* Gaussian distribution can be written as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \qquad x_{\min} \le x \le x_{\max} \tag{1}$$

where μ is the value of the NAS and σ is the value of deviation of the NAS.

From the statistical point of view, every vehicle entering roundabouts can be viewed as an independent event. According to the joint distribution theorem [17], if

driver *A* follows Gaussian distribution $N(\mu_1, \sigma_1^2)$, driver *B* follows $N(\mu_2, \sigma_2^2)$,....., driver *M* follows $N(\mu_m, \sigma_m^2)$, then for independent drivers *A, B,....., M*, the joint distribution of driver *A, B,..., M* follows Gaussian distribution $N(\mu, \sigma^2)$, namely,

$$A + B + ... + M \sim N(\mu, \sigma^2) \quad (2)$$

Concerning the above assumption, driver behavior can be modeled using Gaussian distribution. As such the heterogeneous driver behavior and inconsistent driver behavior can be incorporated by NAS and deviations from it.

2.2 Modeling Vehicles Movement on Urban Streets

The attention was mainly focused into modeling two of three phase traffic flow [18], namely, free flow and synchronized flow. In free flow, a vehicle can drive at its desired speed. In synchronized flow, a vehicle has to follow the vehicle in front. In free flow, speed changes of all vehicles can be assumed to follow a Gaussian distribution. This assumption is based on the fact that the speed changes of an individual vehicle can be approximately seen as a Gaussian distribution, which is described below.

2.2.1 Free Flow Phase

For a vehicle driving between intersections, speed changes are illustrated in Fig. 1, where five stages are involved in our model. In stage A, acceleration of the vehicle is delayed due to physical reasons. In stage B, acceleration of the vehicle increases the speed drastically and leads to the desired speed in stage C. In stage C, speeds randomly fluctuate within the comfortable acceleration/deceleration range [15]. According to the distance between current position and the downstream junction, vehicles are slow down differently in stages D and E. If the vehicle is following a vehicle in front, drivers adjust their speed depending upon speed changes of the preceding vehicle. Speed is adjusted as illustrated in Fig. 1. Probability density of each part follows Gaussian distribution. Update rules of the *n*th vehicle depends on its speed $v_n(t)$ at time step *t*:

1. Speed adjustment

A: $v_n(t) \rightarrow v_n(t) + 1$ B: $v_n(t) \rightarrow v_n(t) + 2$

C: $v_n(t+1) \rightarrow v_n(t) - 1$ with probability p_1 or,
$v_n(t+1) \rightarrow v_n(t) + 1$ with probability p_2 or,
unchanged with probability p_3

D: $v_n(t) \rightarrow v_n(t) - 2$ E: $v_n(t) \rightarrow v_n(t) - 1$

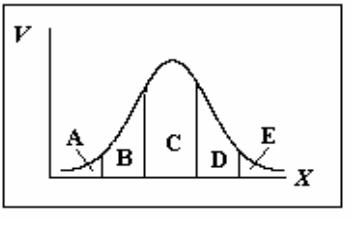


Fig. 1. Speed changes of vehicles in terms of the current position and the distance to the downstream junction. V and X denote the current speed and position, respectively

2.2.2 Synchronized Flow Phase

In synchronized flow, a vehicle has to follow the vehicle in front. Update rules of the *n*th vehicle depends on its position $x_n(t)$, speed $v_n(t)$, and gap (free cells in front) $g_n(t)$ at time step *t*:

1. Speed adjustment

If $g_n(t) < v_n(t)$ then: $v_n(t+1) \rightarrow g_n(t + 2/3)$

This rule is based on the 1.5-second rule. In other words, the vehicle can only drive up to 2/3 of the total distance between the vehicle and the vehicle in front.

Both in free and synchronized flow, the following steps are also implemented to simulate an overall vehicle movement.

2. Randomization

If $v_n(t) > 0$, then the speed of the n-th vehicle is decelerated randomly with probability p_b, i.e. $v_n(t + 1) \rightarrow \max\{0, v_n(t) - 1\}$

3. Vehicle movement

$$x_n(t + 1) \rightarrow x_n(t) + v_n(t + 1)$$

Roundabouts are commonly used in where traffic is not heavy. In other words, the traffic flow that approach a roundabout are normally either free flow or synchronized flow, except at the entrance of a roundabout where queues may form. On the roundabout, the flow can be seen as synchronized flow.

2.3 Modeling Interactions for Vehicles Entry Roundabouts

Vehicles are numbered in the circulatory lane, namely, vehicle $n+1$ precedes vehicle n. Conditions for vehicle k to enter the roundabout are described here. Vehicle n and $n + 1$ are located on the roundabout, while vehicle $n + 1$ has passed the entrance and vehicle n is approaching the entrance. The vehicle k is at the entrance and is waiting for entering the roundabout. Let l_k denote the length of vehicle k, $m_k(t)$ denote NAS of vehicle k, $s_{k,n}(t)$ denote spacing between vehicle k and n at time t. Fig. 2 illustrates the location of vehicles and the topology of the road, and the roundabout. Concerning the above considerations, the following update rules are performed in parallel for all simulated vehicles:

1. Assigning NAS and its deviation for vehicle k according to the probability density of Gaussian distribution.

2. Calculating $s_{k,n}(t)$. If $m_k(t) \leq s_{k,n}(t)$ and $l_k \leq s_{k,n+1}(t)$, the waiting vehicle k can enter the roundabout or if $l_k \leq s_{k,n+1}(t)$, vehicle k can also enter the roundabout, otherwise vehicle k could not enter the roundabout.

3. If vehicle k is waiting for entry, the update rule at each time step is as follows:

$m_k(t) = m_k(t) - \sigma_k$ if a generated random number R ($0 \leq R \leq 1$) < p, p is the predefined number within[0, 1], otherwise $m_k(t) = m_k(t) + \sigma_k$,where $m_k(t)$ and σ_k are NAS (mean) and its deviation of vehicle k.

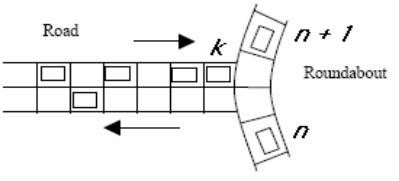


Fig. 2. Schematic diagram of vehicles distribution, a road and a part of the roundabout

3 Experimental Results

The preliminary work is to calibrate vehicle movement on a straight lane. Fig. 3 shows observed single-vehicle movement and its simulation by using the proposed method. We found that when $p_1 = p_2 = 0.3$ and $p_3 = 0.4$, the dual-regime of acceleration and deceleration of our simulation results fits the real behavior of vehicles well, especially in the initial acceleration and final deceleration phases. Probability density of each stage (see in Section 2.2) is assumed to follow Gaussian distribution.

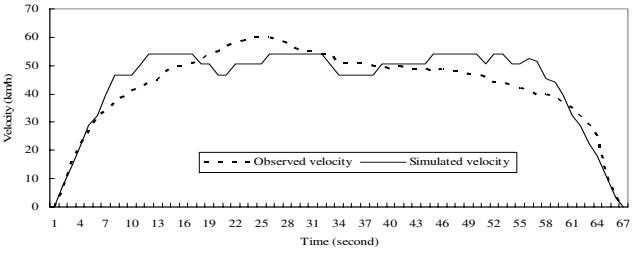


Fig. 3. Simulation of single-vehicle speed between two intersections

Next we apply our model to a case study. Experiments were implemented for 36000 time steps (equivalent to 10 hour) for a street-length of 100 cells on all approaches. The NAS of all drivers ranges within $[x_{min}, x_{max}]$, where x_{min}, x_{max} are taken as 16 and 26 cells in terms of field observation. The mean and deviation of the truncated normal distribution are assumed to be 20 and 2 cells.

To carry out a realistic simulation, many input parameters are required, such as vehicles components, occupied cells, turning rate, arrival rate, etc. We use the data provided in [19] to verify our CA model, where vehicles types and component are given. Table 2 shows comparisons of capacity, delay, and queue length. We can find that capacity has an increase in our model, correspondingly, delay and queue length decrease slightly. For further verifying our model, a comparison with other models (aaSIDRA, UK Linear Regression, HCM 2000, NAASRA 1986) is given in Fig. 4, where capacity of roundabouts computed using our CA model is basically consistent with other models.

Table 2. Comparison capacity, delay, and 95% queue length with our model and [19]. LT=left turning, ST=straight ahead, RT=right turning. Arm 1, 2, 3 and 4 are four roads connection with the roundabout

Road	Vehicles Turning			Vol	Capacity		Delay		95%Queue length	
	LT	ST	RT	.	[19]	Our model	[19]	Our model	[19]	Our model
Arm 1	118	377	150	645	762	775	25	23	10	9.4
Arm 2	88	454	100	642	865	880	15	14	6.86	6.63
Arm 3	107	258	54	419	840	848	8.4	8.2	2.85	2.74
Arm 4	133	586	78	797	963	971	18.9	18	9.8	9.6

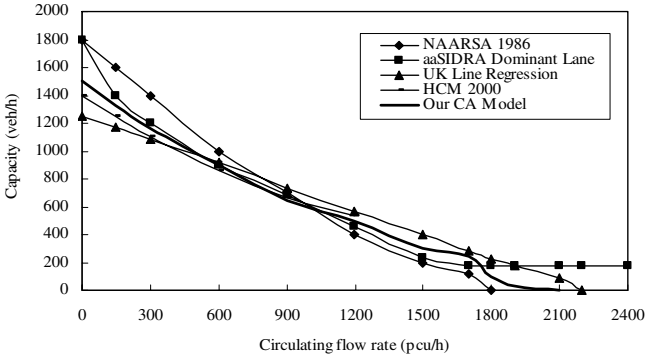


Fig. 4. Comparison of entry capacities estimated by our CA model and other models (the aaSIDRA, TRL (UK) Linear Regression, HCM 2000, NAASRA 1986) [1]

4 Summary

In this paper, we propose a realistic CA model to simulate traffic flow at an urban roundabout. Several important novel features are employed in our model. Firstly, it has been observed that the average headway of car-following is 1.5 seconds in local urban networks and this *1.5-second rule* has been used in modeling the car-following process. Secondly, vehicle movement along urban streets is simulated based on the assumption that speed changes follow a Gaussian distribution. Thirdly, heterogeneous driver behavior and inconsistent driver behavior are modeled using the truncated Gaussian distribution.

Vehicle maneuver on urban roads has been calibrated using field data. The simulation results show that the dual-regime of acceleration and deceleration of the model fits with the real world well.

In order to model a realistic simulation, vehicle arrival rates, turning rates, vehicle types, driver behavior and categorization of speed, etc. are built into our model. The numerical results indicate that the performance (delay and queue length) of roundabouts can be described well.

Acknowledgement

The support of the Massey University Research Fund and the ASIA 2000 Foundation High Education Exchange Programme (HEEP) is a gratefully acknowledged.

References

1. Akçelik, R.: A Roundabout Case Study Comparing Capacity Estimates from Alternative Analytical Models. The 2nd Urban Street Symposium, California, USA, 28-30 July 2003

2. Flannery, A., Datta, T.: Operational performance measures of American roundabouts. *Transportation research Record*, 1572 (1997) 68-75.
3. Troutbeck, R.J.: Background for HCM section on analysis of performance of roundabouts. *Transportation research Record,* 1646 (1998) 54-62.
4. Wang, R.: Modelling Unsignalised traffic Flow with Reference to Urban and Interurban Networks. Doctorate Thesis. Dublin City University (2003)
5. Wang, R., and Ruskin, H.J.: Modeling Traffic Flow at a Single-lane Urban Roundabout, *Computer Physics Communications*, Vol. 147/1-2 (2002) 570-576, Elsevier Science.
6. Campari, E.G., Levi, G., Maniezzo, V.: Cellular automata and roundabout traffic simulation. Proceedings of ACRI 2004: Sixth International Conference on Cellular Automata for Research and Industry, Amsterdam, Netherland, 25-27 October 2004
7. Fouladvand, M.E., Sadjadi, Z. and Shaebani, M.R.: Characteristics of Vehicular Traffic Flow at a Roundabout. Preprints. cond-mat/0309560 (2003)
8. Wang, R., Ruskin, H.J.: Modelling Traffic Flow at a two-lane Roundabout, In: Proceedings of International Conference on Computer Science, Software Engineering, Information Technology, e-Business and Applications, June 5-7, 2003, Rio de Janeiro, Brazil.
9. Chopard, B., Dupuis, A. and Luthi, P.: Traffic and Granular Flow'97, World Scientific (1998) 153-168.
10. Toffoli, T., Margolus, N.: *Cellular Automata Machines--A New Environment for Modelling* http://pm1.bu.edu/~tt/cambook, MIT Press (1987)
11. Nagel, K., Schreckenberg, M.: A cellular automaton model for freeway traffic. J. Phys. I (France) 2 (1992) 2221-2229
12. Simon, P.M., Nagel, K.: Simplified cellular automata model for city traffic. Physical Review E Vol 58, (1998)
13. Barlovic, R., Brockfeld, E., Schreckenberg, M., Schadschneider, A.: Optimal traffic states in a cellular automaton model for city traffic. Traffic and Granular Flow, 2001.10.15 - 2001.10.17, Nagoya University, Japan
14. Neubert, L., Santen, L., Schadschneider, A. and Schreckenberg, M.: Single-vehicle data of highway traffic: A statistical analysis. Phys. Rev. E 60 (1999) 6480.
15. Institute of Transportation Engineers. Traffic Engineering Handbook. (1992)
16. Hays, W.L.: Statistics, the 5th Edition, University of Texas as Austin, Harcourt Brace College Publishers (1994)
17. Kimber, R.M.: The Traffic Capacity of Roundabouts. TRRL Laboratory Report 942. Transportation and Road research Laboratory, Crowthorne, Berkshire, UK (1980)
18. Kerner, B.S. and Rehborn, H.: Experimental Properties of Phase Transitions in Traffic Flow. Phys. Rev. Lett. 79, (1997) 4030–4033
19. http://www.rpi.edu/dept/cits/files/ops.ppt, accessed on 12 October 2004

Probing the Eddies of Dancing Emergence: Complexity and Abstract Painting

Tara Krause

Atelier 1599, 6558 San Haroldo Way, Buena Park, CA 90620 USA
tara@tarakrause.com
http://tarakrause.com

Abstract. Complexity and abstraction provide a fertile frontier by which to express and experience complex systems in art, with integral challenges for abstract painting: Can we create complex art without a computer? What are the simple rules by which to create that art? The author proposes a model of the artist and her materials as a cellular automaton (CA) with eight simple rules. Experiments with the New Kind of Science (NKS) Rule 1599 algorithm, abstract painting and video shorts are discussed with four observations about the plurality of CA visualization, a new abstract visual language that embraces emergence, the discovery of "eddies" of complexity within the paintings, and the resemblance of these eddies to other complex phenomena found in nature and culture. Future exploration merits investigation into the neural basis of art, and experimentation in the synaesthesic experience of complexity.

1 Introduction

Complexity can be generated by software/algorithmic artists and by artists using traditional techniques. This paper explores these seemingly separate paths, which are in fact convergent. Computation does not require artifice, but instead is a natural phenomena. Exciting generative art elicits an almost primal response of recognition. Complexity gives insight into processes in nature and society, and provides a basis for artistic insight.

Over the last 15 years, the concept of complexity has sparked our imagination, not just in science but in art, music and literature as well. Abstract art expert and critic F. V. O'Connor finds inspiration in the new kind of science (NKS): "Let it suffice to state flatly that there is enough dream energy and invincible charm -- enough agency in these physical invisibilities that are part of us -- to inspire a new sense of beauty, a revitalized aesthetic, and those sublime manifestations that are beyond any measurement or ethical judgment, but which can still stun us back to living life to the fullest" [1].

Much of the artistic effort on complexity has been software-based as exemplified by the work of Pegg, Trott, and Tarbell. There have been numerous art and complexity exhibitions. In the net art exhibition catalog of *Abstraction Now*, new media theorist Lev Manovich observed abstract and complexity as a new paradigm for software artists.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 428 – 435, 2005.
© Springer-Verlag Berlin Heidelberg 2005

For a painter, this presents a challenge: Without computers and software-generated algorithms, is complexity in art merely a metaphor, albeit a powerful construct? Can we move beyond complexity-inspired themes to actually create complexity? And if so, what are the simple rules that create complex art? Where is that NKS threshold of complexity where all four classes of behavior are once crossed with further embellishments having little effect?

Certainly there have been artists who have created art that can described as exhibiting NKS Classes 1 (repetitive), 2 (nested) and 3 (random) behavior. Traditional artisan craft is based on repetitive motifs. Abstract Expressionist Jackson Pollack's action paintings and the Japanese woodcuts of waves by Hokusai (1830-1840) have been shown to be fractal in nature. Medici mosaics made by the Cosmati brothers are surprising examples of NKS Class 2 patterns, as are the Byzantine mosaic from the marble pavement in Cosmedin. Earlier Chinese pottery designs appear to be examples of NKS Class 3 behavior

But what about NKS Class 4 complex behavior? This paper attempts to articulate a model in the Liebnizian spirit of the quest for the simple [2], based on Wolfram's standard of visual perception test to determine levels of complexity, and in the tradition of Salvador Dali's exhortation that "to look is to think." One is reminded of early Roman philosopher and dramatist Horace's maxim: '*Pictoribus atque poetis quidlibet audendi semper fuit aequa potestas*[1].' This model proposes that an artist and her materials act as a cellular automaton with simple rules to create complex art. This model is based on observations discovered in a series of both computer experiments and experimentation in the medium of acrylic polymer pigments and emulsions on canvas with a limited historical palette.

As for definitions, emergence is defined as the threshold of complexity at that coalescing moment of self-evidence when new meanings cascade into both the details and the painting as a whole. The term eddy is borrowed from physical oceanography [3] to describe the interesting details that seem to represent a small segment of adhering self-organization within the painting itself. The term primal describes that unconscious immediate response or compulsion, unedited or subjected to conscious thought. The limited palette chosen is grounded in Baroque traditional techniques, as well as pigments found in Upper Paleolithic cave art in Europe. The pigments are restricted to the ochres, siennas, umbers, black and white, with the addition of Indian yellow (traditionally made with the urine of mango leaf fed cows) and alizarin crimson or madder lake.

2 Proposed Model

The proposed model presents an artist and her materials acting as a cellular automaton (CA). In this sense, the art becomes an expression of this complex system. The process cannot by definition be predicted due to irreducibility. As outlined below, there are eight simple rules by which complex art can be created by this CA:

[1] Painters and poets alike have always had license to dare anything.

1) ***An artist and materials act a complex system***. The artist in the model does not impose a planned "cartoon" on her work in the tradition of a Caravaggisti technique, but approaches the painting as an experiment in process rather than pre-determined result. 2) The canvas is upright on the easel at a moderate temperature to ***exploit the reactive medium*** with swipes, glidings, drips and other methods of application. This allows for experimentation in the gravity-driven turbulence and flow of the materials. 3) ***A limited palette*** is used, that is grounded in human artistic tradition. Acrylic paints and polymer emulsions are set up directly on the palette, but only combined on the canvas with a palette knife or brush, not pre-mixed. 4) There is ***a humility of irreducibility*** [4]. The artist must embed the "moi" of the creative process and accept that the brain at criticality acts differently than when imposing solutions. 5) Rather than seeking to represent nature, the artist yields to a ***symbolic nonrepresentational (abstract) visual language*** of sensing complexity through the primal. The intent is to see if patterns of complex behavior can emerge. 6) The artist ***calibrates the painter's eye*** through familiarization to scientific concepts of complexity and physical processes. She learns to recognize NKS classes of behavior through repeated exposure to images of complexity. 7) The artist reaches ***an altered state of consciousness*** through the use of music and dance. She uses her whole body with motion of the lower body driving her strokes and application of painting and medium, versus standing still or sitting. 8) The artist intuitively and rigorously ***seeks the essence*** of her underlying inspiration, asking essential (simple) questions to explore the mechanism of form that moves toward transcendence as Brancusi, Modigliani, Moore, Cecily Brown and Gerhard Richter.

3 Results and Discussion

According to Wolfram, the presence of localized structures that move is a fundamental feature of NKS Class 4 complex behavior. Localized structures are sometimes called gliders in the Game of Life. Within class 4 CAs, systems organize themselves to where definite structures become visible. Some die out, but others persist. These persistent structures transfer information from one part of a class 4 system to another. Sometimes there are collisions between structures producing cascades of new structures.

Code 1599 is a one dimensional, 3 color (2 state) totalistic cellular automata for which no algebraic or Boolean expression exists. This simple rule works by following the rules as shown in Figure 1's rule icon. The values of the cells in an array are updated in discrete steps according to the local rule. This simple program works by having the color of each cell in successive rows be determined by the same simple rule. As a result, a pattern is produced. Interestingly, it takes 8282 steps in evolution to stabilize.

The significance of rule code 1599 is that the patterns seem to follow no definite laws. The structures emerge without anyway to predict them. One can only observe.. Wolfram uses this code to illustrate the principle of computational irreducibility and

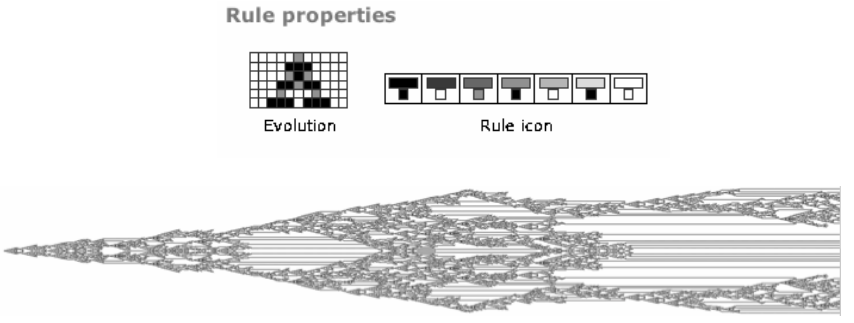


Fig. 1. (Top) NKS Rule 1599 evolution and icon rules. (Bottom) Landscape view of Code 1599 from steps 1 to 8,300 with one black cell as initial condition

its implication for the problems inherent in prediction. The only way to work out the behavior is through observation. One must perform the computation in order to see how the system behaves. This undecidability "makes it ultimately inevitable that such surprises [different forms of behavior emerging] much occur". Wolfram also observes that rule code 1599 seems to show an analog of free will in its computational irreducibility [5]. There is also a possibility that code 1599 is quasi-universal; meaning that like NKS Rule 110, it may have the potential to emulate any other system, though much more investigation is required [6].

A series of computer experiments were conducted to explore the behavior of localized structures in NKS Code 1599 during Wolfram Research Institute's NKS Summer School 2004 at Brown University. An automated search of specified initial conditions of range 1 to 3,000,000 run to step 300 yielded 20,416 files that were sorted into four separate categories of behavior: Dead within 300 steps (11%), Single chain of "lady bug" like structures (19.2%), Single chain of "railroad track" like structures (2%) and those other initial conditions that might create interesting patterns (67.7%). The last category of 13,820 images was visually analyzed. From this analysis, it was observed that the behavior varies across conditions, with only 2 equivalent initial conditions creating the 8,282 steps of evolution. There appeared to be definite structures, some of which were resistant to varying specified initial conditions, and evident patterns of smaller subcomponent elements in both specified and random initial conditions [7].

3.1 Observation 1: There Are a Myriad of Ways to Express Cellular Automata

The experiments were followed by an investigation into visualizing the one-dimensional CA 1599 in three dimensions. These visualizations included various coloration schemes, as well as perspectives such as spinning radius, beaded hanging, and birds-eye view. While interesting, none of these approached the beauty of the CA in the original first dimension.

This raised new questions in terms of the physicality of painting in 2 dimensions. If all the visualizations represented unique but acceptable views, could there not be

other possibilities in expressing the complexity? This opened the degrees of freedom, aptly expressed by Todd Rowland: "Some people describe 1599 mathematically in terms of 1s and 0s. Other people call it cool babushka, and still others teach their children to dance to it" [8]. Moreover, like the Modernist artists struggling with expressing the fourth dimension, contemporary artists are confronted with the phenomena of the first dimension of the computational universe.

There are a myriad of ways of expressing a CA. Stephen Wolfram's use of the cone shell (*conus textile)* is the "classical" example of a 3 dimensional expression of a CA. The cone shell is a 3-D coiled object with the surface as a 2-D pattern generated by a 1-D CA at the living edge of the organism. Another example developed in the experimentation was a tapestry [9]. Birthed of Flatland [10], the tapestry can be conceived as a cellular automata causal network. The digital art image, *NKS_Nasij* (2003), was based upon fragment of a woven golden tapestry (c. 1302) for the emperor Ghazan Khan, grandson of Ghenghis Khan, in Tabriz with Code 1599 superimposed [11]. Considering the cultural complexity of this tapestry's creation and use, this might be an NKS class 4 localized structure that maintains its coherence while moving through its cellular space.

3.2 Observation 2: A New Abstract Visual Vocabulary Embraces Emergence

Yet even with these various visualizations and computer-generated expressions, it was still not evident that emergence as a result of process was possible in painting. Experimentation in the medium of acrylic polymer on canvas was necessary. Earlier efforts in printmaking with experiments in 2003 demonstrated that monotype printing as a reactive medium might produce some evidence of complexity; but due to the medium's heavy reliance on fluid flows of turpentine and inks, it was not conclusive enough.

The acrylic painting experiments used simple rules as outlined in section 2 of this paper for the total of 18 paintings in the *Dancing Emergence* series by Krause. The paintings ranged from 30 inches by 40 inches, to 18 inches to 12 inches. The surprise of the experimentation was that a new abstract visual vocabulary developed. Emergence became an artistic process.

However, there remained the challenge of scaling up the size of the canvas while still maintaining the capacity for emergent patterns. This Richard Serra scale intent became: Can one evoke a complex reaction from the viewer using these images projected on a large screen? How can one create a Rothko-like chapel of a procession of images that evoke complexity?

To work around this challenge, a short film was made that combined Katarina Miljkovic's music compositions of different CA rules (Rules 41 and 1599 along with a Turing Machine) with video footage of the paintings. This experiment demonstrated that this new language was rich enough for further experimentation. Following that up, the *Dancing Emergence* paintings were filmed close-up in motion synchronized to music. In editing the footage, it was discovered that while the paintings cohered as

complex, there were also many individual close-up details of complex "eddies" or localized structures.

3.3 Observation 3: The Eddies Stand on Their Own Terms as Local Structures of Complexity Within the Larger Work

In the approximately 50 minutes of that raw video footage, there were at least 174 examples of images that can be described as complex eddies. Figure 2 below shows the visual evidence of four of these particular eddies. These are still frames of actual details of the larger acrylic polymer and emulsion paintings on canvas.

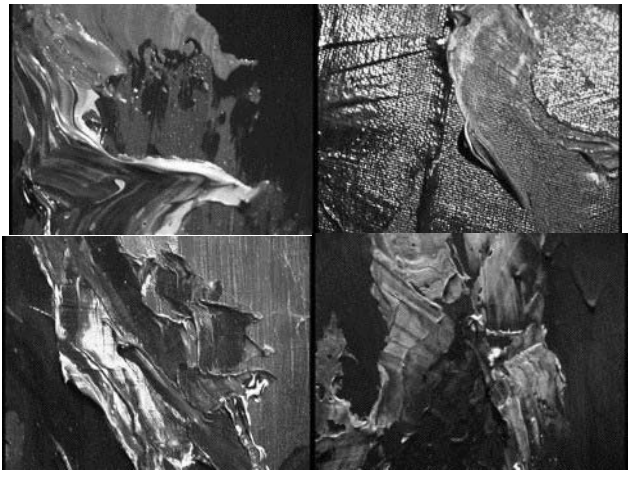


Fig. 2. Details of acrylic polymer and emulsion on canvas. Video stills from *Dancing_Emergence* painting series with the following titles of (clockwise) *DE_17*, *DE_19*, *DE_12*, and *DE_15* (2004)

3.4 Observation 4: The Complexity in the Abstract Paintings Resemble Other Complex Phenomena in Culture

The observation of the paintings' eddies sparked a recognition of similar complex patterns in some Upper Paleolithic microliths. Photographs of Paleolithic rock art seem to appear to show a web of class 4-like finger fluting (intentional engravings) over the initial images. Examples of such are the yellow stallion in the Apse of the Cave of Lascaux (c. 17,000 years ago in the Magdalenian era); another Magdalenian period carved image from the Bases-Pyrénées in France; and the finger tracings of the Gargas and Cosquer caves. One class of Paleolithic finger fluting has been theorized to result from lower body motion. The phenomena of finger flutings in the Rouffignac Cave were to have resulted from moving from the hips with bending, twisting and shifting weight on feet [12]. Some experts have also observed that the size of the handprint signatures in more than twenty caves throughout Italy, France and Spain

point to women shamans and that this indicated a female shamanic role in the spiritual and creative life of the Paleolithic clans.

Even more provocative are the theories of South African San rock art expert Davis Lewis-Williams [13]. His shamanistic art neuropsychological model holds that shamans created these microlithic abstract images from trance phosphenes or entopics (seen by the eye when eyelids are shut) while in an altered state of consciousness (ASC) mode. In Lascaux cave art, Mario Ruspoli stressed that these images have to seen as a whole and mused that perhaps in the flickering of the fire to the rhythm of drums, a shaman in the Lascaux engraved the figures as he told the story before his initiates, that the movements of his hand and the act of drawing combined in its meaning.

Another example of complexity expressed in culture is the Taoist Neo-Confucian taxonomy of *Li* patterns that represent dynamic forms found in nature, considered as laws or principles and expounded on by Chu Hsi during the Sung dynasty (960-1279) and Ch'en Shun. Many of David Wade's categories of *li*, such as breccia and fracture, are strikingly NKSesque if not Class 4 in appearance [14].

These observations raise more questions than answers. What is the neural basis of the artistic creation and perception of such complexity? Is there an interconnection with neural laws of art as theorized by neuroscientist V.S. Ramachandran, such as peak shift, grouping, contrast and isolation [15]? This frontier expands beyond the realm of vision when the question is asked: How does one express and experience cellular automata synaesthesically, where all senses are integrated?

More investigation should be done to address the complexity science underneath these observations: Are these observations a representation of natural processes or a result of a complex process? What is the role of irreducibility? Is the surprise of emergence a result of the process? It does not seem that this could be achieved through an imposed Caravaggisti method. The artist and her materials become the cellular automata.

4 Conclusion

As observed, the creative use of cellular automata provides fertile ground by which complex systems can be expressed and experienced in art, particularly in abstract painting. The artist and her materials function as a computational machine, or cellular automata to create complex art out of simple rules. The patterns in paintings resulting from this CA may resemble complex phenomena in some natural as well as cultural processes. Like the Modernist artists who struggled with expressing Einstein's fourth dimension, contemporary artists are challenged with the phenomena of complexity. New artistic insights from complexity and cellular automata can spark our imaginations, with the songs of the past sung with the rituals of the present to weave the vision and language of the future, where we can sense the kernel of our humanity and environment in the story space of the universe, and where gliders and persistent structures can lead us to transcendence.

Acknowledgements

I would like acknowledge: Drs. Jiri Kroc; composer Katarina Miljkovic of New England Conservatory of Music; Stephen Wolfram and the Wolfram Research Institute team, particularly Todd Rowland, Catherine Boucher, Jason Cawley and Ed Pegg Jr.; and Dale C. Krause of the Marine Science Institute of UCSB.

References

1. F.V. O'Connor: Commentary No.2 (2004) http://members.aol.com/FVOC/comment.html
2. G. Chaitin: Irreducible Complexity in Mathematics (2004) http://www.cz.auckland.ac.nz/CDMTCS/chaitin/latest.html
3. D.C. Krause: The Self-Organized Marine Biological System. In: Soto, L.A, (ed): Ayala-Castanares. Sciencia del Mar y Lmnol. Univ. el NAL. Auton. Mexico (2003)
4. Private communication: J. Cawley, T.Rowland and K. Miljkovic, NKSSS2004 (2004)
5. S.Wolfram: A New Kind of Science. Stephen Wolfram, Champaign (2002)
6. Private Communication: S. Wolfram during NKSSS2004 (2004)
7. T. Krause: Greeting the Muse, NKS Code 1599: Behavior, Motifs & Potentialities for an NKS Way of Art. NKS Summer School 2004. Boston (2004)
8. Private communication: T. Rowland of WRI (2004)
9. Private communication: J. Cawley, T. Roland , R. Philips of WRI and K. Miljkovic of New England Conservatory of Music during NKSSS2004 (2004)
10. E.A. Abbott: A Romance of Many Dimensions (1884)
11. T.Krause: Cellular Automata, Undulating Jellies & Pulsing Bonita. Video, NKS2003, Boston (2003). Color images can be found at http://tarakrause.com
12. K. Sharpe and L. van Gelder. Finger Flutings in Rouffignac Cave, France. (2003)
13. D. Lewis-Williams: Mind in the Cave: Consciousness and the Origins of Arts. WW Norton (2004)
14. D. Wade: Li: Dynamic Form in Nature. Wooden Books, New York (2003)
15. V.S. Ramachandran: A Brief Tour of Human Consciousness. Pi Press, New York (2004)

Enhanced TCP with End-to-End Bandwidth and Loss Differentiation Estimate over Heterogeneous Networks*

Le Tuan Anh and Choong Seon Hong

Computer Engineering Department, Kyung Hee Univerity
1, Seocheon, Giheung, Yongin, Gyeonggi 449-701, Korea
letuanh@networking.khu.ac.kr, cshong@khu.ac.kr

Abstract. The TCP performance degradation over heterogeneous networks is caused by not only network congestion, but also random errors of wireless links. We propose an end-to-end stable accurate rapid bandwidth estimate (SARBE) algorithm reacting appropriately to the end-to-end loss differentiation estimate algorithm (LDED), which can improve the TCP performance over heterogeneous networks without the incipient congestion notifications of the intermediate routers. LDED detects the signal of incipient congestion to lead the sender to enter the congestion avoidance phase opportunely before router's queue overflows. As well as relying distinguishing ability the causes of loss, our algorithm adjusts the packet transmission rate precisely according to the estimated bandwidth after new ACK receiving, fast retransmit or transmission timeout events.

1 Introduction

Wireless and mixed wired-wireless environments are becoming more popular in recent years. The original TCP assumes that every packet loss as an indication of network congestion, which may not apply to heterogeneous networks because packets may be lost by random errors, signal fading or mobile handoff on wireless links. Therefore, in mixed wired and wireless environments, poor performance of TCP is erroneous in behaviors of the congestion avoidance when the packet loss doesn't concern the network congestion.

For the TCP sender, the congestion control probes the available bandwidth of the bottleneck link by continuously increasing the congestion window size (*cwnd*) until reaching the network capacity. When the network congestion is detected by indicating received Duplicate ACKs, the congestion control decreases abundantly to one half of the current *cwnd* setting to the slow start threshold (*ssthresh*). *cwnd* is reset for restarting the slow start phase (SS) until retransmission timer is expired. If packet losses occur by random errors of wireless links before *ssthresh*

* This work was supported by University ITRC Project of MIC. Dr. C.S.Hong is corresponding author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 436–443, 2005.
© Springer-Verlag Berlin Heidelberg 2005

reaches the actual network capacity, *ssthresh* may be obtained a smaller value. Therefore the sending rate is reduced blindly. That is the TCP performance is degraded unreasonably.

In this paper, we are interested in the end-to-end mechanism, in which bandwidth estimate algorithm reacts appropriately incipient congestion signal estimated from the end-to-end loss differentiation algorithm to improve TCP over heterogeneous networks.

The rest of this paper is organized as follows: Section 2 summarizes the related work. Section 3 presents in detail SARBE, and incorporating SARBE and LDED. Simulation results are presented in section 4. Finally, section 5 is for our conclusion.

2 Related Work

There are several approaches proposed for improving TCP performance over wireless networks. They was classified into three classes [2]: the link-layer approach improves wireless link characteristics; the split-connection approach, in which a base station separates the wireless connection from the wired connection, and is responsible for retransmission of packet losses on wireless link; the end-to-end approach, which retains TCP semantics, but requires changes of the protocol stack at either the sender side or the receiver side.

TCP Westwood [4], [5] monitors and averages available bandwidth for every ACK arrived at the sender. The estimated bandwidth is used to control transmission rate via setting *cwnd* and *ssthresh* to the estimated bandwidth after fast retransmission or transmission timeout. Although, the filter of TCP Westwood is complex, it cannot reflect the rapid changes of the network condition. In addition, if ACK packets encounter the network congestion along the backward path, called ACK compression [9], TCP Westwood overestimates the available bandwidth.

The end-to-end loss differentiation proposals categorize the packet losses explicitly through different estimate without any support from the intermediate routers, such as Flip Flop [7], Vegas [11] and Non Congestion Packet Loss Detection (NCPLD) [12]. They are based on the TCP state variables and information of ACKs to estimate the reason of packet losses. NCPLD categorizes the nature of the error by detecting the knee point of the throughput-load bend.

The Vegas predictor measures the lowest Round Trip Time ($RTTmin$) during the TCP connection and computes the expected throughput ($cwnd/RTTmin$). When the sender receives ACK, it computes the actual throughput ($cwnd/RTT$). [11] defined extra packets between two thresholds α and β in the network as

$$D_{Vegas} = RTT_{\min} \times \left(\frac{cwnd}{RTT_{\min}} - \frac{cwnd}{RTT} \right) \tag{1}$$

If $D_{Vegas} \geq \beta$, the Vegas predictor detects the network becoming congestion. Otherwise, if $D_{Vegas} \leq \alpha$, there are more available bandwidth for connection. In

the other hand, the network state is kept the same as in the last estimate when $\alpha < D_{Vegas} < \beta$.

The parameters $\alpha = 1$ and $\beta = 1$ is not accurate that proved in [11]. The authors of [10] then showed that the predictor achieves the highest accuracy if $\alpha = 1$ and $\beta = 3$.

3 Proposal

3.1 Available Bandwidth Estimate

In stable accurate rapid bandwidth estimate (SARBE) algorithm. The ACKs sending time intervals are used to compute the available bandwidth of the forward path via the timestamp of ACK. The estimate of the forward path is not be affected by ACK compression that results in overestimate.

To estimate the current bandwidth by observing the pattern of bandwidth for consecutive packets, this can be written as

$$Bw_k = \frac{L_k}{ts_k - ts_{k-1}} \tag{2}$$

where L_k is the amount of data acknowledged by the kth ACK, ts_k is timestamp of the kth ACK; ts_{k-1} is the timestamp of the previous ACK arrived at the sender.

We used the stability-based filter [8] which is similar to the EWMA filter, except using a measure function of the samples' large variance to dynamically change the gain in the EWMA filter. After computing the bandwidth sample Bw_k from (2), the stability-based filter can be expressed in the recursive form

$$U_k = \beta U_{k-1} + (1 - \beta) \mid Bw_k - Bw_{k-1} \mid$$

$$U_{max} = max(U_{k-N}, ..., U_{k-1}, U_k)$$

$$\alpha = \frac{U_k}{U_{max}} \tag{3}$$

$$eBw_k = \alpha \cdot eBw_{k-1} + (1 - \alpha)Bw_k \tag{4}$$

where U_k is the network instability computed in (2) by EWMA filter with gain β, β was found to be 0.8 in our simulations; $Umax$ is the largest network instability observed among the last N instabilities ($N = 8$ in our simulations); and eBw_k is the estimated smoothed bandwidth, eBw_{k-1} is the previous estimate and the gain α is computed as (3) when the bandwidth samples vary largely.

We evaluate the stability, accurateness and rapidity of SARBE. The simulation network scenario is depicted in Fig. 1. We used an FTP over TCP and an UDP-based CBR background load with the same packet size of 1000 bytes. The CBR rate varies according to time as the dotted line in Fig 2(a).

The result is shown in Fig. 2(a); TCP Westwood is very slow to obtain the available bandwidth changes. By contrast, SARBE reaches the persistent bandwidth changes rapidly, which closely follow the available bandwidth changes.

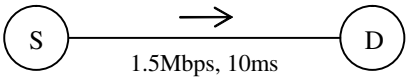


Fig. 1. Single bottleneck link

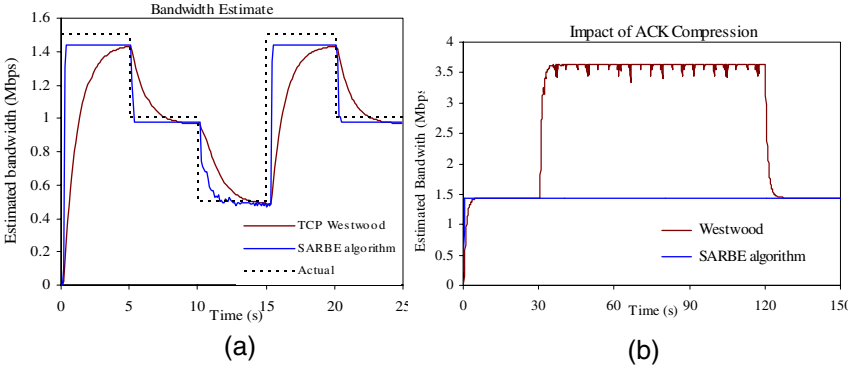


Fig. 2. (a) Comparison of Bandwidth estimate algorithms, (b) Overestimated bandwidth of TCP Westwood

This is due to adaptability of dynamic changes of gain when the bandwidth samples vary largely.

To investigate the impact of ACK compression on estimate, we used the network scenario as Fig. 1 and supplemented a traffic load FTP in the reverse direction. The traffic load FTP was started at time 30s and ended at 120s for 150s simulation time. In this interval, Westwood estimates over 2 Mbps more than SARBE, which is quite near the actual available bandwidth, as in Fig 2(b).

The TCP's *ssthresh* represents the probed network bandwidth; while the above estimated bandwidth value also represents the current available bandwidth of forward path. Consequently, we have to transform the estimated value into equivalent size of the congestion window for updating *ssthresh*. [5] proposed the interrelation of the estimated bandwidth with the optimal congestion window size ($oCwnd$) as

$$oCwnd = \frac{eBw \cdot RTT_{min}}{Seg_size}$$

where RTT_{min} is the lowest Round Trip Time, Seg_size is the length of the TCP segment.

3.2 Enhanced TCP with Incorporating SARBE and LDEA

In our design, we propose a new scheme by incorporating SARBE and LDEA. For LDEA, we apply the equation (1) to detect the network becoming congestion for every ACK arrived at the sender. Accordingly, the sender can distinguish the packet losses caused due to congestion from those caused due to random

errors of wireless links. And then, relying distinguishing the causes of losses, our scheme adjusts the packet transmission rate precisely according to the estimated bandwidth after new ACK receiving, fast retransmit or transmission timeout event occurs.

The pseudo code of our algorithm is presented following.

A. Algorithm after receiving ACK or Duplicate ACKs

```
if (ACK is received)
  /* calling the loss differentiation estimate algorithm */
  if (cwnd < ssthresh and isIncipientCongestion== true)
    ssthresh = oCwnd;
  endif
endif
if ( n DupACKs are received)
 ssthresh = oCwnd;
 /* the packet loss is caused by congesion */
 if (isIncipientCongestion == true)
   if (cwnd > ssthresh )
     cwnd = ssthresh;
   endif
 else /* the packet loss is not caused by congesion */
     /* keeping the current cwnd */
 endif
endif
```

Whenever the sender receives a new ACK with incipient congestion, the congestion control updates *ssthresh* to *oCwnd* during the slow start phase (SS). Setting precisely *ssthresh* to the available bandwidth of bottleneck link leads the sender to enter the congestion avoidance phase (CA) opportunely before router's buffer overflow.

When Duplicate ACKs are received, *ssthresh* is set to *oCwnd*. If the packet loss is caused by the network congestion, the congestion control should restart the CA phase during the CA phase. Otherwise, it keeps the current *cwnd*.

B. Algorithm after timeout expiration

```
if ( retransmission timer expires)
  ssthresh = oCwnd;
  cwnd = 1;
endif
```

If the sender is triggered by the retransmission timeout event due to the heavy network congestion or very high bit-error rate of wireless link, the congestion control sets *ssthresh* to *oCwnd* and then sets *cwnd* to one for restarting the SS phase.

4 Simulation Results

All of our simulations were run by the NS-2 simulation network tool [6]. We used the recent Westwood module NS-2 [3] for comparison.

4.1 Effectiveness

The simulation was run in a simple hybrid environment, shown in Fig. 3(a). The topology includes the bottleneck capacity of 5 Mbps, one-way propagation delay of 50 ms, the buffer capacity equal to the pipe size, and a wireless link.

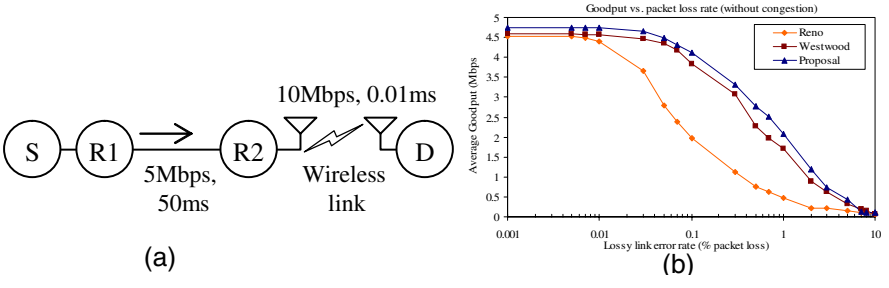


Fig. 3. (a) Single bottleneck link; (b) Comparison of Bandwidth estimate algorithms

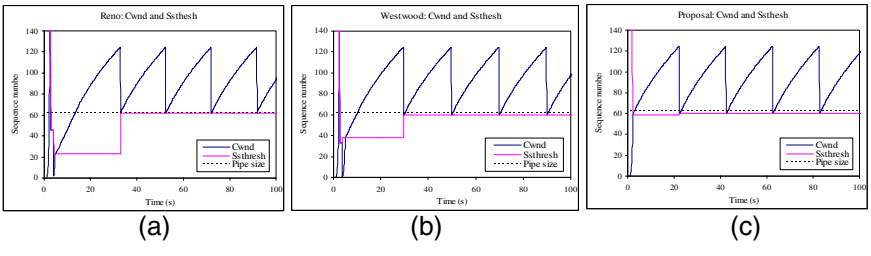


Fig. 4. (a) *Cwnd* and *ssthresh* of Reno, (b) *Cwnd* and *ssthresh* of TCP Westwood, (c) *Cwnd* and *ssthresh* of the proposed TCP in the absence of random errors

We evaluate TCP performance in the lossy link environment. The simulation was performed on one FTP in 100s with the packet size of 1000 bytes, the wireless link random errors ranging from 0.001% to 10% packet loss. In Fig 3(b), for any random error rate, the goodput of the proposed TCP is better than other versions. Particularly, at 1% wireless link packet loss rate, the proposal can achieve better performance than TCP Reno and Westwood by 76.6% and 17.9%, respectively.

Outperforming of the proposal at any random error rate less than 0.001% can be explained by the different behavior of the three protocols shown in Fig 4. In Fig 4(a), (b), at the beginning of the TCP connections, TCP Reno and TCP

Westwood increase exponentially their *cwnd*s to probe the network capacity. Until the router's buffer was overflowed, and then the retransmission timeout events occur, they set *ssthresh* to one-half of the current *cwnd* for TCP Reno, to the estimated bandwidth for TCP Westwood, and restart the SS phase. In contrary, relying the incipient congestion signal of LDEA, the proposed TCP can update *ssthresh* to the estimated bandwidth in the SS phase. This leads the sender to enter the CA phase opportunely before router's queue overflows, shown in Fig 4(c).

4.2 Fairness

The fairness of TCP depicts the fair share ability of the bottleneck bandwidth with multiple connections of the same TCP version. The fairness index is used to assess the convergence of TCP. It was proposed in [1] as following

$$fi = \frac{(\sum_{i=1}^{n} x_i)^2}{n(\sum_{i=1}^{n} x_i^2)}$$

where x_i is the throughput of the ith TCP connection, n is the number TCP connections considered in simulation. The fairness index has a range from $1/n$ to 1.0, with 1.0 indicating fair bandwidth allocation.

Using the same scenario as Fig. 3(a) with ten same TCP connections, we simulated the different TCP versions individually. The buffer capacity of bottleneck link is equal to the pipe size. The comparison result is shown in Fig. 5(a). The proposed TCP, TCP Reno and TCP Westwood can achieve high fairness index.

4.3 Friendliness

The friendliness of TCP implies fair bandwidth sharing with the existing TCP versions. We considered a total of ten connections mixing the proposed TCP with TCP Reno and Westwood at 1% packet loss rate of the wireless link. The x-axis of Fig. 5(b) represents the number of TCP Reno, Westwood connections; the remaining connections used in the proposed TCP. In Fig. 5(b), the proposal proves the coexistent ability with the TCP Reno, but outdoes in goodput.

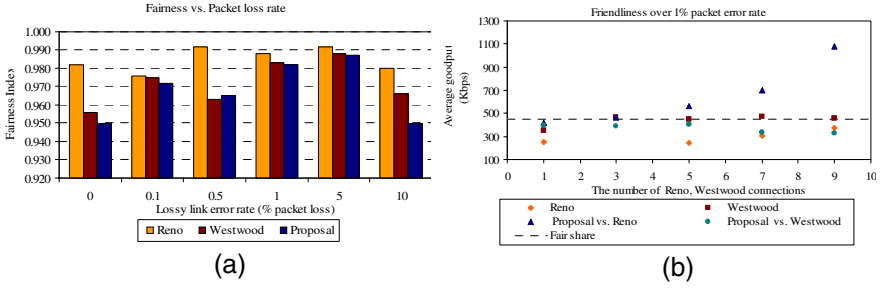


Fig. 5. (a) Fairness vs. packet loss rate; (b) Friendliness of TCP Reno and Westwood compared with the proposal, respectively, over 1% packet loss of wireless link

5 Conclusion

By incorporating the stable accurate rapid bandwidth estimator and the loss differentiation estimator, our proposal can react appropriately to the packet losses in heterogeneous networks, where the losses are caused by either network congestion or random errors of wireless links. LDED detects the network becoming congestion to lead the sender to enter the CA phase opportunely before router's queue overflows. As well as relying on distinguishing ability the causes of loss, our algorithm adjusts the packet transmission rate precisely according to the estimated bandwidth obtained from SARBE, after new ACK receiving, fast retransmit or transmission timeout events.

References

1. R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC, Rep.TR-301, 1984.
2. H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," IEEE/ACM Trans. Networking, vol. 5, no. 6, pp. 756769, 1997.
3. TCP Westwood - Modules for NS-2 [Online]. Available: http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_ns2/tcp-westwood-ns2.html, 2004.
4. S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi,and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in Proc. ACM MobiCom 2001, Roma, Italy, pp. 287297, July 2001.
5. S. Mascolo, C. Casetti, M. Gerla, and S.S. Lee, M. Sanadidi, "TCP Westwood: Congestion Control with Faster Recovery," UCLA CS Tech. Report. #200017, 2000.
6. NS-2 network simulator [Online]. Available: http://www.isi.edu/nsnam/, 2004.
7. D. Barman and I. Matta, "Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks," Boston University Technical Report, 2002.
8. M. Kim and B. D. Noble, "SANE: stable agile network estimation," Technical Report CSE-TR-432-00, University of Michigan, Department of Electrical Engineering and Computer Science, Ann Arbor, MI, August 2000.
9. L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic," Proc. SIGCOMM Symp. Comm. Architectures and Protocols, pp. 133-147, Sept. 1991.
10. S. Bregni, D. Caratti, and F. Martigon, "Enhanced Loss Differentiation Algorithms for Use in TCP Sources over Heterogeneous Wireless Networks," in IEEE Global Communications Conference, Globecom 2003, Dec. 2003.
11. S. Biaz and N. H. Vaidya, "Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result," Seventh International Conference on Computer Communications and Networks (IC3N), New Orleans, Oct. 1998.
12. N.K.G. Samaraweera, "Non-Congestion Packet Loss Detection for TCP Error Recovery using Wireless Links," IEE Proceedings Communications, volume 146 (4), pages 222-230, August 1999.

Content-Aware Automatic QoS Provisioning for UPnP AV-Based Multimedia Services over Wireless LANs

Yeali S. Sun[1], Chang-Ching Yan [1], and Meng Chang Chen[2]

[1] Dept. of Information Management, National Taiwan University, Taipei, Taiwan
sunny@im.ntu.edu.tw
[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan
mcc@iis.sinica.edu.tw

Abstract. With the advent of wireless and mobile devices, wireless communications technology is enjoying its fastest growth period in history. It has been greatly affecting the way we live and work. Two main challenges remain that prevent the wide-spread adoption of digital media distribution using wireless technology such as IEEE 802.11 Wi-fi LANs at home are plug-and-play (zero-configuration) and Quality of Service (QoS). The UPnP AV (audio/video) technology is an emerging multimedia session initiation/control protocol promoted by Intel and Microsoft for multimedia content delivery services in home networks. In this paper, we propose a new scheme to address the above two issues. By performing Layer-7 content classification, inspection and automatic resource allocation and configuration, the scheme provides transparent QoS guarantees to UPnP AV multimedia streaming applications over wireless LANs such as in home and office environments. The execution of these operations is automatic and completely transparent to end users. Users are free of complex QoS configuration and multimedia service technology details. The scheme requires no changes of any existing protocols and network interface cards. A Linux-based wireless home gateway router is also prototyped. The performance results measured from the testbed show that the system achieves the goals of providing home users zero configuration and transparent QoS guarantees for multimedia streaming over wireless home networks.

1 Introduction

Digital and wireless networking will become prevailing technologies to provide a solid foundation for distributing entertainment content at home. Two main challenges remain that prevent the wide-spread adoption of digital media distribution using wireless technology such as IEEE 802.11 wi-fi LANs at home are plug-and-play (zero configuration) and Quality of Service (QoS). Unlike IT professions, no home users can tolerate complex configuration and manipulation of consumer electronics devices. Image you have to teach your grandparents and kids about link sharing policy, packet classification rules and various VoIP and MPEG compression algorithms and bit rates so they will be able to configure the wireless access points/gateway router to guarantee the QoS of a movie or mp3 streaming in their

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 444 – 452, 2005.
© Springer-Verlag Berlin Heidelberg 2005

home network. Thus, providing transparent QoS service is essential to the success of home networking. That is, the networking devices are smart enough to automatically detect the establishment of multimedia sessions (or connections) among all the traffic transported in a wireless home network, as well as to guarantee the QoS of these sessions.

The UPnP (Universal Plug and Play) [1] is a technology proposed by a number of vendors including Microsoft and Intel for peer-to-peer networking of intelligent appliances, wireless devices and personal computers. The UPnP AV (Audio/Video) architecture [2] is a sequel of UPnP which is specially designed to support AV content delivery between networked devices. The UPnP AV technology is expected to have widespread support from the industry to incorporate it into devices/equipment in home networking in the future. While the current UPnP AV architecture reaches a state of readiness, how to transparently guarantee the quality of service of AV content transfer in a wireless home or office environment remains to be resolved. It truly relies on the network equipment as an essential component to do wireless channel resource management and provide a transparent, convenient environment to assure users have comfortable experience of enjoying multimedia services with QoS guarantees at home. In a home network, we envision there will be a diverse variety of applications such as telephony, video and data traffic *share* the same wireless bandwidth.

In this paper, we present a new scheme, based on content-aware packet classification and automatic configuration, to provide transparent QoS guarantees to UPnP AV sessions over wireless LANs. The key ideas in our scheme are as follows:

- Innovative content-aware packet classification of AV signaling/control messages to learn the establishment and release of AV sessions.
- Real-time content inspection to extract QoS requirements of AV session (e.g., application type, bit rate, identifications of end points, etc.).
- Integrating the scheme with underlying network QoS mechanisms to transparently and automatically perform bandwidth reservation and packet classifier configuration.

To achieve automatic QoS provision, one must know the control protocols used by the multimedia sessions. Our approach is to intercept and classify UPnP AV session control messages, perform Layer-7 content inspection to retrieve QoS requirements from the UPnP AV XML-based messages, automatically configure the underlying network QoS mechanisms and packet classifier to assure the wireless channel access of the session based on the requirements (such as the bit rate, and transport information of the communicating devices (e.g., IP address, port number, etc.)). The scheme requires no changes of any existing standards (e.g., IEEE 802.11 MAC and content delivery applications). It is completely transparent to the users.

The rest of the paper is organized as follows. In Section 2, we briefly describe the UPnP AV architecture. In Section 3, we describe the proposed scheme in details, including system architecture and the content classification/inspection algorithms. We have implemented the proposed scheme on a Linux-based QoS wireless gateway router with packet classification/scheduling/queueing mechanisms supported in the

kernel. In section 4, the performance results measured from the prototype system and the tested are presented. Finally, the conclusion is given in Section 5.

2 UPnP AV Architecture

As shown in Figure 1, there are three components in the UPnP AV architecture: *MediaServer* [3], *MediaRenderer* [4], and *Control Point*. The architecture defines the interaction between the UPnP AV Control Points and UPnP AV devices. The "MediaServer" is the source of the media content and allows AV Control Point to browse the content items that are available for users to enjoy. Three services have been defined in the MediaServer: Content Directory Service [5], Connection Manager Service [6] and AV Transport Service [7]. A MediaRenderer device obtains and renders audio/video content from a MediaServer device via network. There are three services defined in MediaRenderer: Rendering Control Service [8], Connection Manager Service, and AV Transport Service. The Control Point is a device that provides user interface for users to select desired content available on MediaServer devices to MediaRenderer devices of choice. It manages the operations of both MediaServer and MediaRenderer by sending instruction messages to the devices. The AV content streaming is however, directly from MediaServer to MediaRenderer.

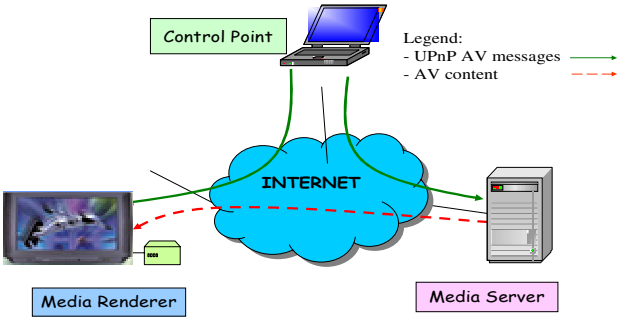


Fig. 1. The UPnP AV architecture

Figure 2 shows the message flows in the UPnP AV architecture.

- ContentDirectoryService::Browse() action allows Control Points to obtain detailed information of the media files that MediaServer provides such as the name, artist, transfer protocol and media format that MediaServer supports.
- The ConnectionManager::GetProtocolInfo() action allows Control Points to learn the transfer protocol and media format that MediaRenderer supports.
- The ConnectionManager::PrepareForConnection() action is used by Control Point to inform MediaServer and MediaRenderer that an AV session is about to be established and the selected transfer protocol and media format for the session.

- AVTransport::SetAVTransportURI() action is used to inform MediaRenderer of the URI of the AV media file to be rendered.

The UPnP AV architecture allows MediaServer and MediaRenderer to choose any transfer protocol for content delivery. In the case of HTTP[10], after receiving AVTransport::Play(), MediaRenderer will send a HTTP GET message to MediaServer. When the AV content transfer is stopped, MediaRenderer will notify Control Point by sending a notification message. For RTSP[11]/RTP[12], after receiving AVTransport::SetAVTransportURI(), MediaRenderer will send a RTSP SETUP message to MediaServer. Upon receiving AVTransport::Play(), MediaRenderer will send a RTSP PLAY message to MediaServer. Finally, MediaRenderer sends a RTSP TEARDOWN to terminate the RTSP session.

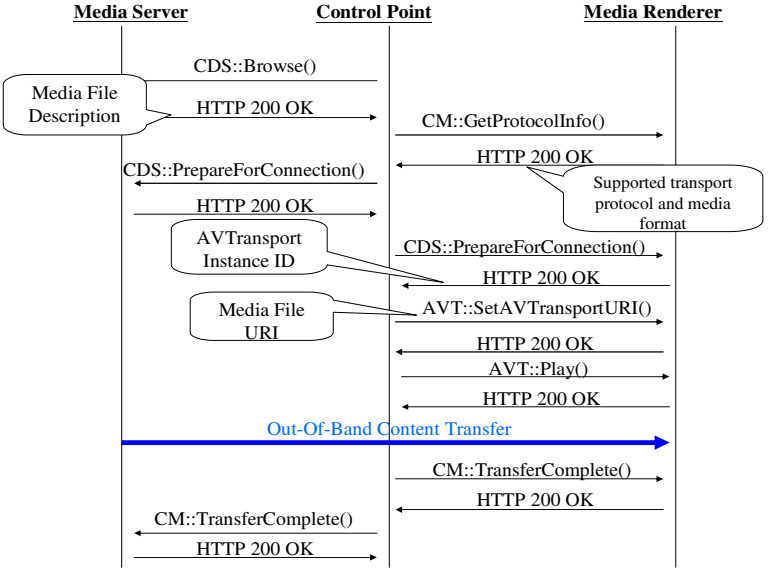


Fig. 2. The procedure and messages exchange to set up an UPnP AV session

3 Transparent QoS Provisioned UPnP AV Multimedia System

3.1 Wireless Home Network

In this paper, we consider a wireless home network as shown in Figure 3. The Control Point module is located in a wireless gateway router. The MediaServer device is attached to the home network through a wired or wireless link. One or more MediaRenderer devices such as TV can be dispersed in different locations in the house and are connected to the wireless gateway router through wireless links such as

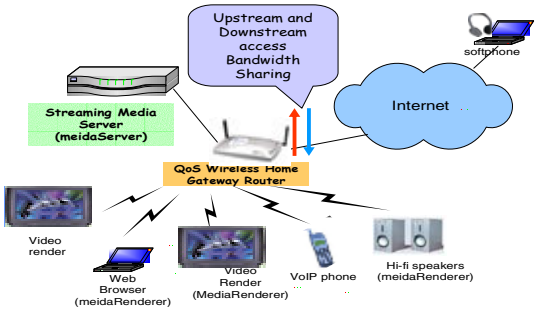


Fig. 3. Multimedia services over a wireless home network

IEEE 802.11. The typical AV content distributions are such as an MPEG-4 movie streaming from a personal video recorder with storage (MediaServer) to a TV (MediaRenderer) and a MP3 music from a computer (MediaServer) to hi-fi stereo speakers (MediaRenderer). In these scenarios, the interactions (control and data) between control points, content servers and rendering devices are all via the wireless gateway. Our objective is to design a software module inside the wireless gateway router for two purposes. First, since all the traffic will pass through the gateway router, the module will monitor and detect the exchange of the UPnP AV signaling messages for AV session establishment and release. It will then perform content inspection to retrieve QoS-relevant information from relevant messages.

Second, once obtaining the information, the module in the wireless gateway will automatically configure the underlying QoS mechanisms to allocate necessary bandwidth to assure the transport quality of the audio/video session. The execution of these operations is automatic and completely transparent to end users. In other words, users do not need to do any configuration of any of these UPnP AV devices and the wireless gateway router. Users are free of complex QoS configuration and multimedia service technology details. Our software provides a convenient QoS-guaranteed wireless multimedia digital home entertainment environment. It requires no changes to any existing standards and network interface cards.

3.2 System Architecture

The system architecture is shown in Figure 4. The *Packet Type Classifier* filters out the UPnP AV messages subject to content inspection. The *Packet Content Parser* is responsible for parsing message content to retrieve necessary information for automatic QoS configuration. The *Session Manager* and *MediaInfo Manager* manage two major data structures - the QoS session descriptors and media information descriptors - for the active AV sessions in the system, respectively. The *FSM* module implements the finite state machines (FSM) of the transport protocols supported in the system. The *FSMs* are the procedures for content classification and inspection of the UPnP AV messages and the messages of the supported transport protocols to manage AV sessions. The *QoS Manager* interacts with the kernel QoS modules to make and release bandwidth reservation.

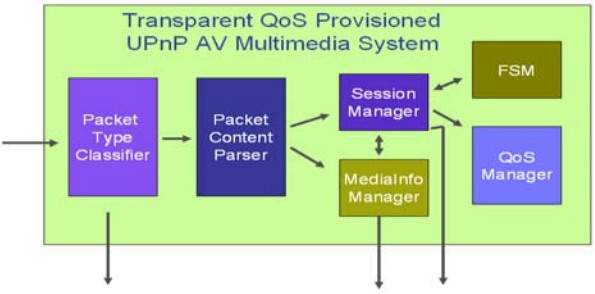
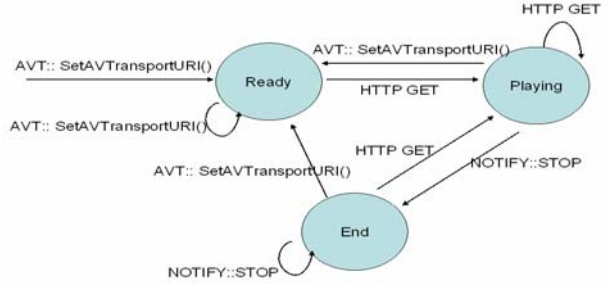


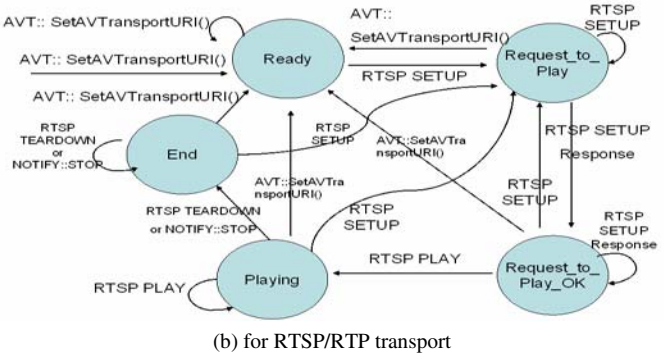
Fig. 4. The architecture of the transparent QoS provisioned UPnP AV multimedia system

3.3 The Finite State Machines for Content-Aware Packet Classification and Inspection

Two transport protocols - HTTP and RTSP/RTP – are currently supported in the proposed system. The corresponding finite state machines (FSM) are shown in Figure 5.



(a) for HTTP transport



(b) for RTSP/RTP transport

Fig. 5. The finite state machines of the UPnP-AV content-aware classification and inspection

3.4 QoS Information Extraction and Algorithm

Figure 6 presents the UPnP AV message content inspection algorithm used to retrieve necessary information for automatic QoS configuration and provisioning. The UPnP AV messages are in the HTTP/XML format. The algorithm interacts with the underlying QoS mechanisms (e.g., packet classifier, packet scheduler and queue manager) to perform real-time resource allocation to assure the content delivery performance of the session in the wireless channel (both upstream and downstream).

```
switch (UPnP message)
{
  case (CDS::BrowseResponse):
            Search for media file(s) to playback;
            Create a Media Information Descriptor for each media file;
  case (AVT::SetAVTransportURI):
            Parse the message to retrieve QoS relevant data;
            Create a QoS Session descriptor;
  case (HTTP GET):
            Search the QoS Session descriptorDB that matches the IP address and URI in the message;
            If found, pass QoS parameters (MediaRendererPort, MediaServerIP, MediaServerPort) to
               QoS Manager to make bandwidth reservation;
  case (RTSP SETUP):
            Search the QoS Session descriptorDB that matches the IP address and URI;
            If found, pass QoS parameters (RTSP Session Identifier, CSeq) to QoS Manager to make
               bandwidth reservation;
  case (RTSP SETUP response):
            Parse the message to get MediaRendererPort and MediaServerPort;
            Update the QoS Session descriptor of the session;
  case (RTSP  PLAY):
            Search the QoS Session descriptorDB that matches the IP address and RTSP Session Identifier;
            If found, pass QoS parameters to QoS Manager to make bandwidth reservation;
  case (RTSP TEARDOWN):
            Search the QoS Session descriptorDB that matches the IP address and RTSP Session Identifier;
            If found, pass session information to QoS Manager to release bandwidth reservation;
            Delete this session's Session descriptor from the QoS Session descriptorDB;
  case (NOTIFY::STOP):
            Search the QoS Session descriptorDB that matches the IP address and AVTranspont Instance ID;
            If found, pass session information to QoS Manager to release bandwidth reservation;
            Delete this session's Session descriptor from the QoS Session descriptorDB;
}
```

Fig. 6. The content inspection algorithm of UPnP AV messages for automatic QoS configuration and provisioning

4 Performance Evaluations

We have implemented the proposed transparent QoS-provisioned system as a Linux kernel module on an IEEE 802.11b wireless gateway router [13] (an Intel Pentium 3 PC with Prism2 wireless card and the Linux kernel is 2.4.19). The testbed is similar to the configuration as shown in Figure 3. The AV content flow is from the MediaServer to the MediaRenderer via HTTP. Figure 7 and Figure 8 show the throughput and delay performances of two UPnP AV movie streaming sessions measured from the tested when sharing the wireless channel with a 5Mbps UDP flow.

With the QoS system enabled, each UPnP AV session automatically receives the required bandwidth allocation and the content rendering is smooth and of good quality.

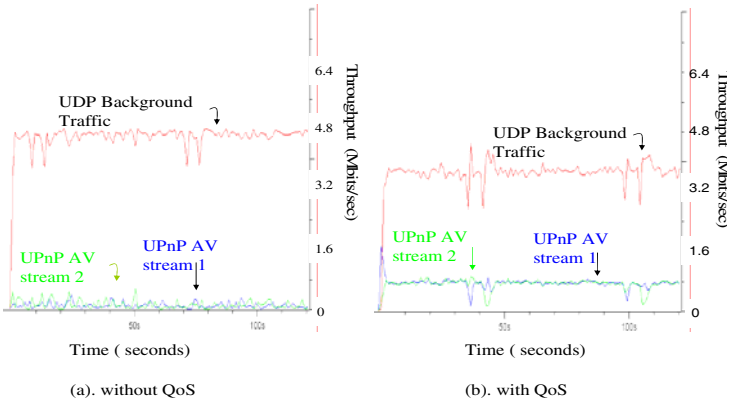


Fig. 7. Throughput performance of the UPnP AV streams with a UDP background traffic

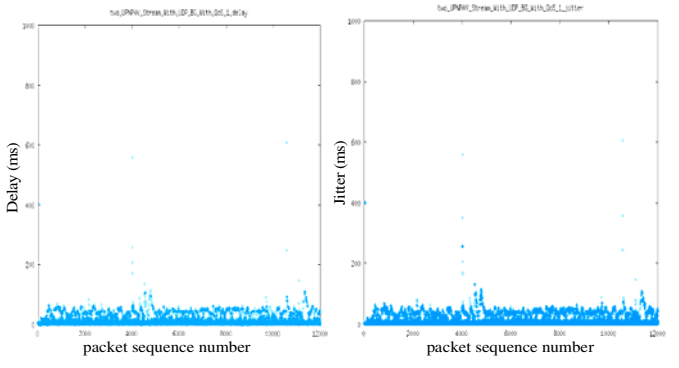


Fig. 8. Delay and jitter performances of the UPnP AV streams with content-aware automatic QoS-provisioned system enabled

5 Conclusion

In a home network, there will be a diverse variety of applications such as telephony, video and data traffic *share* the same wireless bandwidth. Unlike IT professions, no home users can tolerate complex configuration and manipulation of consumer electronics devices. In this paper, we present the design and implementation of a content-aware automatic QoS provisioning system. The goal is to implement such system on a wireless home gateway device to provide home users with a transparent, convenient environment to assure their comfortable experience of enjoying

multimedia services at home. The signaling or control architecture and protocol considered in the system is the UPnP AV architecture which is an emerging industrial standard for digital home networks. The proposed scheme is implemented on a Linux-based wireless AP. The performance results measured from the testbed show that our system can correctly identify all UPnP AV sessions and detect the start and termination of data transfer of each AV session. By integrated with the underlying network QoS mechanisms, the wireless home gateway router can provide QoS-guaranteed transmission service to multimedia applications over the wireless LAN.

References

[1] The UPnP™ Forum, "UPnP Device Architecture 1.0," May 2003.
[2] John Ritchie and Thomas Kuehnel, "UPnP AV Architecture:0.83," June 2002.
[3] John Ritchie, "MediaServer:1 Device Template Version 1.01," June 2002.
[4] John Ritchie, "MediaRenderer:1 Device Template Version 1.01," June 2002.
[5] Kirt Debique, Tatsuya Igarashi, Sho Kou, et al., "ContentDirectory:1 Service Template Version 1.01," June 2002.
[6] Shannon Chan, Alec Dara-Abrams, Mike Dawson, et al., "ConnectionManager:1 Service Template Version 1.01," June 2002
[7] Larry Buerk, Jean Moonen, Dale Sather et al., "AVTransport:1 Service Template Version 1.01," June 2002.
[8] Sho Kou, Takashi Matsui, Jean Moonen et al., "RenderingControl:1 Service Template Version 1.01 1.01," June 2002.
[9] T. Berners-Lee, R.Fielding and H.Frystyk, "Hypertext Transfer Protocol – HTTP/1.0," RFC 1945, May 1996.
[10] R. Fielding, J. Gettys, J.Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, 1999.
[11] H. Schulzrinne, A. Rao and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, April 1998.
[12] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Read-Time Applications," RFC 1889, January 1996.
[13] Yeali S. Sun and J. F. Lee, "Policy-based QoS Management in NBEN – Differentiated Services Provisioning," TANET2000, October 2002.

Simulation Framework for Wireless Internet Access Networks

Hyoung-Kee Choi and Jitae Shin

The School of Information and Communication Engineering,
Sungkyunkwan University,
Suwon, Korea 440-746
{hkchoi, jtshin}@ece.skku.ac.kr

Abstract. In many fields of engineering and science, researchers and engineers use computers to simulate natural phenomena rather than conducting experiments involving the real system. With the power of today's computers, simulation provides an easy way of predicting the performance of a prospective system or comparing several alternatives at the system design stage. In this paper, we present a simulation framework specifically designed for wireless Internet access networks. This simulation framework is designed for the three protocol layers, HTTP, TCP/IP and the link-layer protocol, and can be configured independently in each of these layers. The time-driven nature of the framework allows us to observe the diurnal changes of the system in the simulation, which in turn makes it possible to evaluate the statistical properties of the system.

1 Introduction

In recent years, Internet technology has emerged as the major driving force behind new developments in the area of telecommunication networks. The volume of packet data traffic has increased at extreme rates. In order to meet these changing traffic patterns, more and more network operators are adapting their strategies and are planning to migrate to IP-based backbone networks. Meanwhile, mobile networks face a similar trend of exponential traffic increase and growing importance to users. Recently, in some countries, such as the Republic of Korea, the number of mobile subscriptions has exceeded the number of fixed lines.

The combination of both developments, the growth of the Internet and the success of mobile networks, suggests that the next trend will be an increasing demand for mobile access to Internet applications. It is therefore increasingly important that mobile radio networks support these applications in an efficient manner. Thus, the mobile radio systems currently under development include support for packet data services. For instance, General Packet Radio Service (GPRS) is the wireless packet data service on the Global System for Mobile (GSM). New wireless packet data services have also been introduced for the Wireless Local Loop (WLL) and CDMA2000 systems, which operate in parallel to the existing wireless circuit voice services.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 453–460, 2005.
© Springer-Verlag Berlin Heidelberg 2005

When the new wireless packet data systems were designed, a simulation was used to predict the performance of the prospective system or to compare several alternatives [1],[2],[3],[5],[7],[8]. Hence, it is important to have a reliable simulation framework which accurately predicts the performance of the future system. A number of simulation frameworks have been introduced for network systems. Of these, *ns-2* (Network Simulator), ssf (Scalable Simulation Framework) and Opnet are the most popular tools for network simulation. These simulators are generally considered to be the best tools available and the models on which they are based have been thoroughly validated by a number of user groups. However, in the case of wireless network simulations, the choice of one of these tools might not be optimal, since they were all developed for general purpose networking simulations.

In a previous study, we reported the development of a behavioral Web traffic model [4]. Since then, this model has been used in at least three studies to evaluate the performance of wireless networks [5],[7],[8]. However, these studies did not include certain key points such as how the model was adapted to satisfy the requirements of the study, how the different stacks of the protocol interacted with one another to produce the right simulation results, and so on. In this paper, we provide more detailed and up to date information on our model and the associated simulation framework, which are designed to be used in wireless internet access networks.

2 Proposed Model

We characterize the HTTP and TCP layers as well as an underneath link-layer protocol of interest based upon the typical transactions of the individual protocol. We examined the transactions associated with the retrieval of a single Web page and selected a set of primary parameters and secondary parameters in each layer. The primary parameters can be used to define the Web traffic model, and the secondary parameters help to understand the behavior of the Web traffic. In the following discussion, we use **boldface** to indicate a parameter.

2.1 HTTP Model

We characterize Web traffic based upon a typical transaction of a Web page in HTTP, as shown in Fig. 1. The number of objects refers to the total number of objects in a Web page. Nine objects are in the Web page shown in Fig. 1. **Number of in-line objects** is eight, and there is also one main object. We count only those objects that need to be downloaded. If an object is cached, the browser uses the object in the local cache after validating the object.

A new Web page is generated immediately after the expiration of the viewing period. HTTP alternates between the ON and OFF states, in which the ON state represents the activity of the Web page and the OFF state represents the silent time after all the objects in the Web page have been retrieved. The durations of the ON state and the OFF state correspond, respectively, to **on-time** and

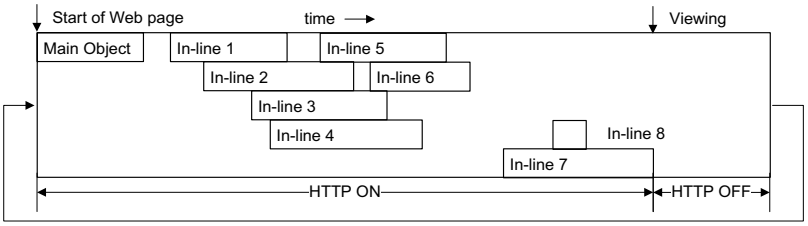


Fig. 1. A typical transaction of a Web page in HTTP

viewing time. The ON state can be further split into the successive TCP connections used to deliver individual objects. The five rows in Fig. 1 represent distinct connections.

We refer to the connections in rows 1, 2 and 4 as keep-alive connections, because multiple objects are delivered in one connection. In Fig. 1, we consider in-line objects 1, 5, 6 and 8 as being delivered on the keep-alive connections. **Keep-alive ratio** is calculated by dividing the number of objects delivered on the keep-alive connections by the number of objects (the number of in-line objects plus one) for a Web page. **Keep-alive ratio** in the above example is 0.44 (4/9). The inactive period between in-line objects 6 and 7 is denoted as **in-line inactive time**. In particular, the inactive period between the main object and in-line object 1 is required in order to parse the HTML code, and is denoted as **parsing time**. A Web server identifies a requested object based on the HTTP request header. We denote the size of the HTTP request header as **request size**.

2.2 TCP Connection Model

We characterize TCP based upon a typical transaction of a TCP connection used to retrieve a Web object, as shown in Fig. 2. At the beginning of the connection, the client and server exchange control segments (SYN) to synchronize with each other. It takes a single round-trip time (RTT) to exchange SYN segments (see the 0th period in Fig. 2). Once the synchronization process is completed, the client sends a request for an object (REQ in Fig. 2) to the Web server.

TCP alternates between inactive and active periods of transmitting data segments. After a TCP sender transmits an entire window-size worth of data segments in a burst, it pauses until the first ACK corresponding to the burst returns (see first and second periods in Fig. 2). This is because the window size is still too small for the pipe to be completely filled. Then, TCP starts the next burst after adjusting the window size. As the window size increases, the inactive period decreases.

Based upon the bursts in a TCP connection, we characterize TCP by: (1) defining the period between the starts of adjacent bursts and (2) measuring the number of data segments transmitted and the time spent in this period. Let us denote the period between the starts of adjacent bursts as a window epoch or

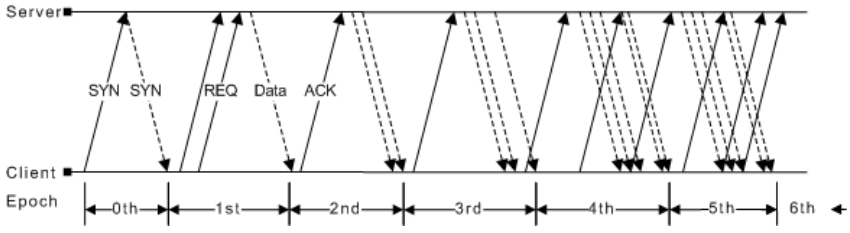


Fig. 2. A typical transaction of a TCP connection transferring a Web object

an epoch. Let us denote the number of data segments transmitted and the time spent in an epoch as **number of segments in an epoch** and **epoch time**, respectively.

2.3 Link-Layer Protocol Model

A number of wireless protocols can be combined in this proposed simulation framework, in order to measure their performance. To help understand how one can use the proposed simulation framework, we illustrate one typical simulation procedure with a MAC protocol in a satellite network. This MAC protocol was developed by the author and the details of the MAC protocol can be found elsewhere [5].

Remote stations under consideration are Web clients using the TCP stack as their primary transport protocol. The remote stations send a request to the hub to gain access to the physical channel. The hub receives the request and schedules it according to the centralized priority reservation (CPR) protocol. The physical channel is divided into a number of forward and return links. Forward links are generally larger in capacity than return links, as they carry more data to the remote terminals than the return link carries to the hub.

When an IP packet is generated at the remote station, it is passed to the MAC layer. Upon receiving a packet, the MAC protocol divides it into data frames. Before transmitting data frames, the remote station sends a request for transmission to the hub. This request is sent on a contention-basis. A collision may occur between different remote stations having a request to make at the same time. Provided that no collision occurs for the request, the hub acknowledges the request immediately. If a collision occurs when the request is being sent, the remote station will not receive an acknowledgment from the hub after the round-trip delay and will then attempt to retransmit the request. A contention resolution algorithm (CRA) is used to resolve the collision.

3 Traffic Generation

The complete model of Web traffic is a combination of three models: the HTTP model, the TCP model and the model of the link-layer protocol of interest. These three models interact with each other to form a total model that encompasses all

Web traffic. The statistics of the parameters and their probability distributions can be found elsewhere [4],[5],[6].

3.1 HTTP Layer

Our model in the HTTP layer simulates an ON/OFF source. At the beginning, the traffic corresponding to the main object is generated and is delayed for the period of **parsing time**. During this period, the Web browser fetches the main object and parses number of in-line objects as well as the page layout. The HTTP model, however, generates the value of **number of in-line objects** from the best-fit distribution and waits for the expiration of **parsing time**.

After the start of one in-line object, there is a delay before the start of the next. The first in-line object starts after the expiration of **parsing time**. The second in-line object does not wait until the first in-line object finishes, but starts one **in-line inter-arrival time** after the start of the first. Subsequent in-line objects continue to start until the number of in-line objects started equals **number of in-line objects**. In the model, depending upon **in-line object size** and **in-line inter-arrival time**, the number of outstanding connections will vary. Frequently, **in-line inter-arrival time** is less than the duration of the connection, which is mainly determined by **in-line object size. Hence, the model indirectly simulates the parallel downloading of in-line objects**. After all of the objects have been transmitted, the model is silent for a period of time corresponding to **viewing time**. After the expiration of this time period, the model starts to generate a new Web page.

The Web caching model influences the final model through main object size and in-line object size. Due to frequent changes that they undergo, the main objects are fetched most of the time, rather than being cached. The HTTP object size becomes zero, except for that of the main object if it is destined to be cached. Otherwise, the sizes of both HTTP object types are generated from the distribution.

3.2 TCP Layer

For the complete model of Web traffic, the TCP model relies on the HTTP model to obtain essential information regarding a connection. The TCP model obtains the object size from the HTTP model. In addition, a real connection can be a keep-alive connection, delivering more than one object. The HTTP model determines whether a given connection is a keep-alive connection and the elapsed time between objects.

At the beginning of a connection, the client in the model exchanges a SYN segment with the server, mimicking the three-way handshaking procedure. After the three-way handshaking procedure, the client enters the first epoch by sending a request segment. At this point, the HTTP model informs the TCP model of **request size**, **main object size** and **in-inline object size**. The model calculates the total number of segments in the connection by dividing the object size by an MSS of 1,460 bytes. The number of segments in the first epoch and

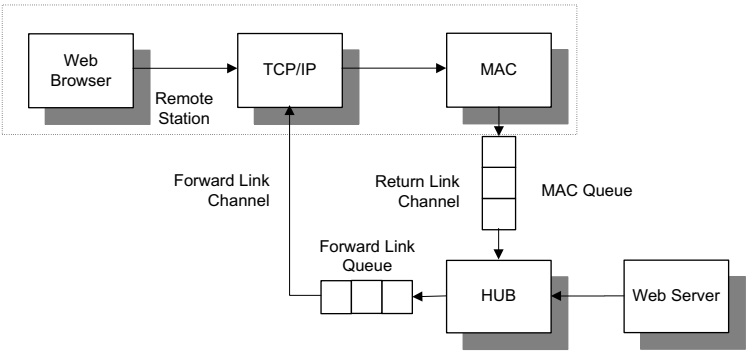


Fig. 3. Software Implantation of the link-layer model

the time the first epoch lasts is determined from the distributions of number of segments and epoch time given epoch number one.

In a given epoch, the TCP model generates a burst of segments followed by an idle period until the next epoch starts. At the end of each epoch, the model checks whether the server has transmitted enough segments for the given object size. The model proceeds to the next epoch, as long as the cumulative number of transmitted segments is less than the total number of segments. At the last epoch, when the model finishes downloading the current object, the HTTP model informs the TCP model of its decision as to whether the connection should remain open for the delivery of the next object or should be closed.

3.3 Link Layer

Although it is the TCP model that determines when to send packets, it is the link-layer model that determines how to send packets in the wireless channel. For this particular illustration of the simulation, we set the return-link bandwidth and forward-link bandwidth to 32 kbps and 45 Mbps, respectively. The RTT between the remote station and the hub through the satellite was set to 0.5 seconds. The number of remote stations is varied, in order to observe the changes in throughput and load induced by changing the number of stations.

Once a packet is available at the remote station, the TCP model informs the link-layer model of the packet size. The link-layer model segments the packet into frames of the proper size, as determined at the protocol design stage. For individual frames, the link-layer model simulates the MAC protocol, in order to obtain access to the wireless channel. Once the remote station has secured the right to access the wireless channel, it may deliver the frame to the hub.

Due to the time-driven nature of the framework, the clock in the simulation increments with the fixed size of an interval. This time-driven simulation enables us to observe the longitudinal changes of the system. In this way, it is a lot easier to evaluate the statistical properties of the system being examined. A great deal of care needs to be taken in deciding the granularity of the interval. With a small

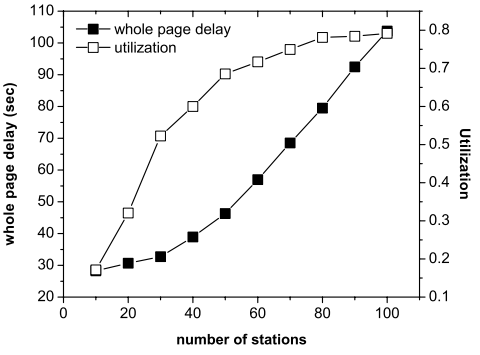


Fig. 4. Delay vs. number of stations and channel utilization vs. number of stations

granularity, the system can be observed in detail, but the simulation is likely to take too long. In our design, the incrementing interval is equivalent to the time delay required for transmitting one frame over the wireless channel.

Frames arriving at the hub are reassembled in order to be transmitted over the wire line toward the Web server. Since our primary interest is the system behavior in the wireless channel, we do not specifically implement the mechanics of the wire line in the simulation. Instead, the mechanics of the wire line are replaced by the statistics collected in the trace. In this way, we can reduce the complexity of the simulation, without sacrificing the accuracy.

The procedure used for the return-link channel is also applied to the forward-link channel, with two exceptions, namely the bandwidths are different and the MAC protocol does not exist in this channel. The packets arriving at the remote station are processed in the link-layer protocol and then passed successively to the TCP and HTTP layers. From the perspective of the simulation, this is the end of a single round trip for the packet. The next round trip is initiated by the dynamics of the upper layer protocols. However, the procedure used for the next round trip is the repetition of the current round trip.

4 Discussion

Among the numerous measurements that can be obtained from the simulation, the most interesting ones are the delay and the throughput with different loads. Fig. 4 shows a plot of the delay and the throughput versus the load. As the number of users (load) increases, saturation starts to occur when the number of users is in the range of 40 to 50. In addition to this result, we can also extrapolate that the delay varies linearly with the number of remote stations. The feedback nature of the Web browsing application (HTTP) limits the number of outstanding packets, so that at no point does the simulation collapse; the response time just becomes slower.

For simplicity, we assume that the capacity in the forward-link channel is infinite, so that we may not implement the mechanics of the forward-link channel in the simulation exclusively. We refer to this type of simulation as an open-loop simulation. The reason for this is that the bandwidth of the forward-link channel is so big that outstanding frames would not experience any delay in the forward-link queue shown in Fig. 3. As a result, this assumption does not affect the overall performance of the simulation, rather the complexity of the simulation decreases significantly.

It is preferable to obtain a quick response from a given simulation. At the same time, a stead-state response is also required. However, this is not an easy task, because the simulation experiences a transient state in the beginning. For instance, the queues in the simulation are completely empty in the beginning and take a certain amount of time until they reach a steady state. Any conclusions drawn from the transient state may undermine the accuracy of the simulation. Hence, the simulation must be long enough to allow a steady state to be reached.

5 Conclusion

We proposed a simulation framework for wireless Internet access networks. This framework was developed in three layers, the HTTP, TCP/IP and Link-layer protocols. Because of this layered structure, one can not only examine the behavior of the system being examined at each layer, but also observe the overall behavior of the system within the combined layers. Because the proposed framework has a time-driven nature, one can observe the diurnal changes of the system at regular intervals, which is very important when it comes to evaluating the statistical properties of the system.

References

1. Kalden, R. and et al.: Wireless Internet Access Based on GPRS, *IEEE Personal Communcation Magazine*, April 2000.
2. Brasche, G. and Walke, B.: Concepts, Services, and Protocols of the New GSM Phase 2+ GPRS, *IEEE Communcation Magazine*, August 1997.
3. Cai, J. and Goodman, D.: General Packet Radio Service in GSM, *IEEE Communcation Magazine*, October 1997.
4. Choi, H. and Limb, J. O.: A behavioral model of Web Traffic, In *Proceedings of the IEEE ICNP '99*, October 1999.
5. Choi, H. and et al.: Interactive Web service via satellite to the home, *IEEE Communcation Magazine*, March 2001.
6. Choi, H. and Copeland, J. A.: Modeling the behavior of TCP in Web traffic, In *Proceedings of the ICOIN '05*, January 2005.
7. Mohangy, B. and et al.: Application Layer Capacity of the CDMA2000 1xEV Wireless Access System, In *Proceedings of the World Wireless Congress*, May 2002.
8. Staehle, D. and et al.: QoS of Internet Access with GPRS, *Wireless Network*, May 2003.

WDM: An Energy-Efficient Multi-hop Routing Algorithm for Wireless Sensor Networks

Zheng Zengwei[1,2], Wu Zhaohui[1], Lin Huaizhong[1], and Zheng Kougen[1]

[1] College of Computer Science, Zhejiang University,
310027 Hangzhou, China
{Zhengzw, Wzh, Linhz, Zkg}@cs.zju.edu.cn
[2] City College, Zhejiang University,
310015 Hangzhou, China
Zhengzw@zucc.edu.cn

Abstract. As a new technique, one characteristic of wireless sensor networks (WSNs) is their limited system lifetime. Therefore, it is more important to save energy and proportion energy consumption. This paper presents a weight-directed based multi-hop routing algorithm for WSNs. This algorithm can transfer data quickly to goal sensor node using the directional information and RWVs (route weight value) of sensor nodes as well as balance energy consumption of all sensor nodes. Detailed simulations of sensor network environments indicate that this algorithm improves energy efficiency and proportions energy consumption of all sensor nodes to extend network system lifetime, and routes data quickly in comparison to the flooding algorithm.

1 Introduction

Recent advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of wireless sensor networks (WSNs) consisting of a large number of low-cost, low-power, multi-functional sensor nodes which are small in size and communicate untethered in short distances. As a new technique of implementing ubiquitous computing [1][2], WSNs can be used in many aspects in the coming future, such as military battlefield surveillance, patient health monitoring [3], bio-environment monitoring [4] and industrial process control. Since the sensor nodes are often inaccessible in most applications and supported by battery, the lifetime of a wireless sensor network depends on the lifetime of the power resources of the sensor nodes. Hence, WSNs have one characteristic different from traditional Ad Hoc networks, i.e. their system lifetime is limited.

The characteristic of limited lifetime indicates that energy is a very scarce resource for the sensor systems and requires a new sensor network with low energy consumption in order to extend the lifetime of the sensors for the duration of a particular mission. Since the main goal in designing conventional Ad hoc networks is providing high quality of service, conventional wireless network protocols for Ad Hoc networks are not well suitable for WSNs. Furthermore, requirements of designing routing algorithm for WSNs is different from those for traditional Ad Hoc networks, i.e. it needs more energy savings. Therefore, it is significant to study new routing algorithms for WSNs.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 461 – 467, 2005.
© Springer-Verlag Berlin Heidelberg 2005

As multi-hop routing shortens communication distance, short transmission range will reduce packet collisions, enable channel reuse in different regions of a wireless sensor network, lower energy consumption of sensor nodes, and prolong the lifetime of sensor nodes. Hence, multi-hop routing idea is suitable for WSNs. Existing multi-hop routing algorithms, such as flooding, gossiping [7] and directed diffusion [5][6], have several advantages and disadvantages. Flooding can perform simply, route data quickly, and it does not require costly topology maintenance and complex route discovery algorithms. However, when it broadcasts packets to its neighbors, implosion problem will be generated, and a large number of data flow and subsequently channel congestion, and communication overheads may happen. Therefore, its energy-efficiency is wondrously low. A derivation of flooding is gossiping [7] in which nodes do not broadcast but send the incoming packets to a randomly selected neighbor. Although the implosion phenomenon can be avoided by the copy of a message at any sensor node, it takes long time to propagate the message to goal sensor node (sink node) because it does not use the directional information to route data quickly to objective node. Directed diffusion [5][6] is a data-centric and application-aware routing protocol. All sensor nodes in the directed diffusion-based network are application-aware, which enables diffusion to achieve energy savings by selecting empirically good paths and by caching and processing data in the network. The directional information is utilized in this paradigm, and this protocol is mainly used in the observer-initiated application model [8]. However, it is necessary to design a routing algorithm in the continuous application model [8]. Therefore, this paper proposes a new multi-hop routing algorithm (WDM algorithm, Weight-Directed based Multi-hop routing algorithm) to meet the requirements of the continuous application model [8]. This approach can transfer data quickly to goal sensor node, i.e. sink node (See Fig. 1), using the directional information and RWVs of sensor nodes (RWV definition shown in Equation (2)), and balance energy consumption of all sensor nodes.

The remainder of this paper is organized as follows: The details of WDM algorithm are given in section 2. The simulation results of analyses are illustrated in section 3. Finally, conclusions are made and some suggestions for future work are proposed.

2 WDM Algorithm

WSN is presented as undirected graph G = (V, E), where V is the set of all sensor nodes, i.e. $V = \{V_1, V_2, ..., V_n\}$, $N = \{1, 2, ..., n\}$, and E is the set of edges in the network defined as follows:

$$E = \left\{ (V_i, V_j) \mid d(V_i, V_j) \le R_0, V_i, V_j \in V, i, j \in N, i \ne j \right\} \tag{1}$$

Where $d(V_i, V_j)$ is the distance between the neighbor nodes V_i and V_j, R_0 is described as one-hop transmission range.

A number of definitions related to Equation (1) are given as follows:

(1) $V_i.hop_{min}$ is the minimal hops from sensor node V_i to sink node, $V_i \in V$.
(2) V_i.UP_Node is the neighbor node of V_i whose hop_{min} value is equal to $V_i.hop_{min} - 1$. A set composed of these nodes is named as V_i.UpNodeSet.

(3) V_i.Par_Node is the neighbor node of V_i whose hop_{min} value is equal to $V_i.hop_{min}$. A set made up of these nodes is marked as V_i.ParNodeSet.
(4) V_i.Down_Node is the neighbor node of V_i whose hop_{min} value is equal to $V_i.hop_{min}+1$. A set composed of these nodes is named as V_i.DownNodeSet.
(5) V_i.NBNodeSet is the union set of the above three sets.
(6) After system initialization, sink node first broadcasts route query packet to all sensor nodes. Then, each sensor node can gain route information, compute its hop_{min} value, and save its each neighbor node's hop_{min} and residual energy value E_r in its cache.

In the case of invariable topology, once a source senor node V_s senses a data packet of a particular mission, if a sensor node V_i have gained the message and $V_s.hop_{min}$ value from node V_s, it will first compute route weight value (RWV) of its each neighbor nodes when choosing next hop node, which is defined as follows:

$$V_i.RWV(V_k) = \left(\frac{V_s.hop_{\min}}{V_k.hop_{\min}}\right)^{\alpha} \frac{V_k.E_r}{V_k.E_0} \tag{2}$$

Where α is the effect factor of route direction, $V_i \in V$, $V_k \in V_i$.NBNodeSet, $V_k.E_0$ is the initial energy value of V_k. If each sensor node's initial energy is assumed as same value, $V_k.E_0$ is abbreviated as E_0.

Then, V_i selects the neighbor node V_k whose RWV is maximal, and also send the message and $V_s.hop_{min}$ value to node V_k. Subsequently, V_k chooses the next hop node and transfers the packet to it until the message is sent to sink node.

When V_i node has sent data packet to the node V_k, V_i updates the remainder energy of its neighbor V_k. Specific computational approach is defined as follows:

$$V_k.E_r^{'} = V_k.E_r - V_k.E_{ex} - V_k.E_{tx} \tag{3}$$

Where $V_k.E_r^{'}$ is the new residual energy value of node V_k; $V_k.E_r$ is the old residual energy value of node V_k; $V_k.E_{ex}$ is the energy consumption value when node V_k receiving one data packet; $V_k.E_{tx}$ is the energy consumption value when node V_k transferring one data packet.

At the same time, V_k also modifies the residual energy value of its neighbor V_i. The estimate method is described as follows:

$$V_i.E_r^{'} = V_i.E_r - V_i.E_{ex} - V_i.E_{tx} - V_i.E_c \tag{4}$$

Where $V_i.E_c$ is the energy consumption value of node V_i computing and selecting next hop node; the meanings of $V_i.E_r^{'}$, $V_i.E_r$, $V_i.E_{ex}$, and $V_i.E_{tx}$ are the same as above.

3 Performance Evaluation

3.1 Simulation Environment and Testing Criterion

A square region of $100\times100m^2$ has been generated and 160 sensor nodes are placed in the network randomly (See Fig. 1). All nodes start with an initial energy of 10J. The details of the sensor node's energy consumption model are shown in [9][10][11]. Data packet size is 525bytes, α is equal to 1, the maximum distance of one hop is 15m. It is assumed that a random node can sense one data packet of a particular mission at intervals of 1ms, and each message is finally routed to sink node.

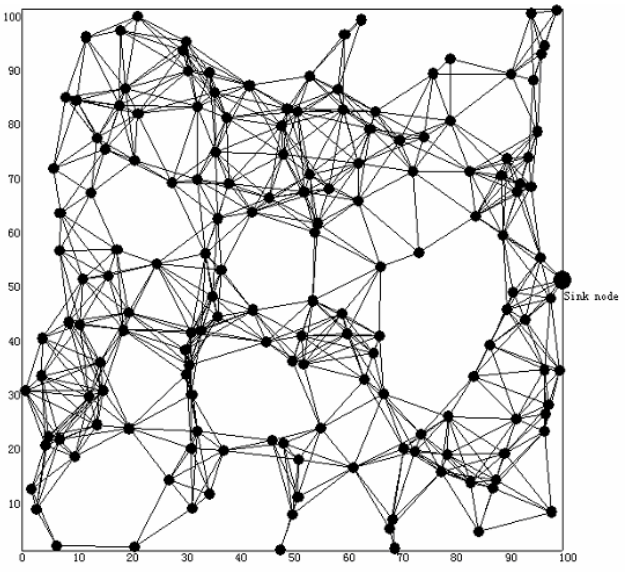


Fig. 1. Sensor nodes scattered in a wireless sensor network (sink node (100,50))

In order to analyze WDM algorithm performance, flooding algorithm is utilized to compare with WDM algorithm, and the following performance metrics are used:

(A) Average hops per data packet routed from source node to sink node (AHPD): this metric shows time delay of routing data and reflects whether it takes long time of algorithm to transfer data to sink node. A method of computing the metric is shown as follows:

$$AHPD = \frac{\int_0^t Data_hops(x)dx}{\int_0^t Data_num(x)dx} \tag{5}$$

Where Data_hops(t) is a linear function of hops with respect to time variable t; Data_num(t) is a linear function of data packet number about time variable t.

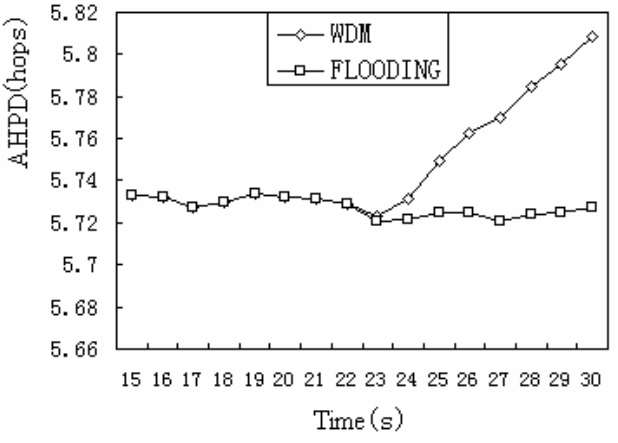


Fig. 2. Comparison of time delay of routing data between WDM and Flooding

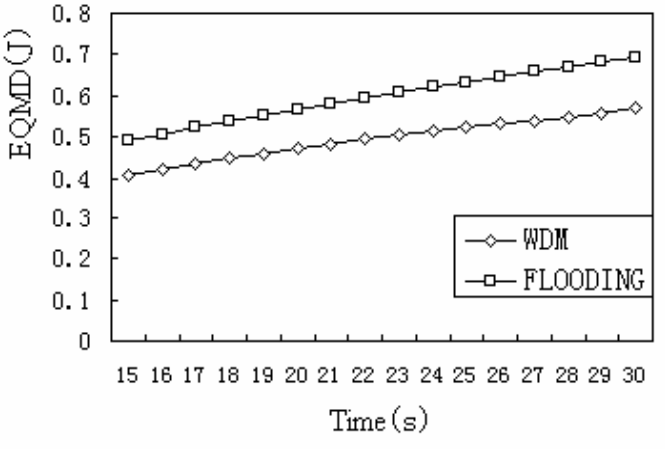


Fig. 3. Comparison of evenness of energy dissipated between WDM and Flooding

(B) Network energy quadratic mean deviation (EQMD): this metric indicates evenness of network energy dissipated of sensor nodes. A method of calculating this metric is listed as follows:

$$EQMD = \sqrt{\sum_{j}(\sum_{i} V_i.E_r / All_nodes_Num - V_j.E_r)^2} \tag{6}$$

Where All_nodes_Num is the total number of sensor nodes in the network; $V_i.E_r$ is defined as residual energy value of one sensor node at the time.

(C) Average energy dissipated per data packet routed from source node to sink node (AEPD): this metric reflects energy costs of transferring data packets to sink node and shows energy-efficiency of algorithm. Combined with metric (B), it indicates the

ability of algorithm extending system lifetime. A means of computing this metric is shown as follows:

$$AEPD = \frac{\sum_i V_i.E_0 - \sum_i V_i.E_r(t)}{\int_0^t Data_num(x)\,dx} \tag{7}$$

Where $V_i.E_r(t)$ is the remainder energy value of node V_i at one time t; the definitions of $Vi.E_0$ and Data_num(t) are listed as above.

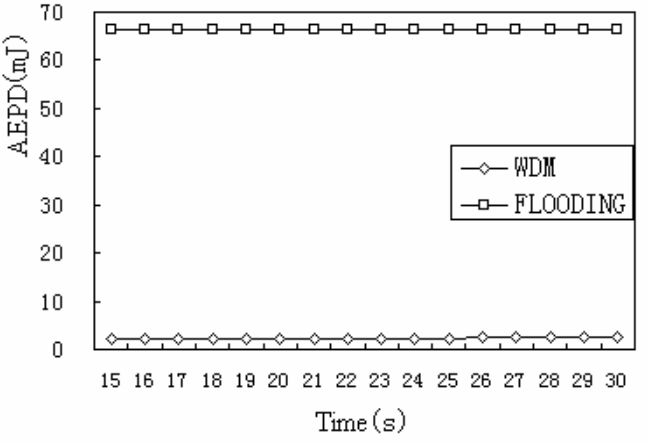


Fig. 4. Comparison of energy-efficiency between WDM and Flooding

3.2 Result Discussion

Firstly, in order to test time delay of routing data packets, simulation is performed with metric (A) and the results are shown in Fig. 2. It is found that the WDM algorithm can also transfer data quickly to sink node though it is slightly slower than the flooding algorithm because the latter is the quickest among all multi-hop routing algorithms for sensor networks.

Then, the evenness of dissipated network energy is also conducted with metric (B), as listed in Fig. 3. It is shown that two algorithms have the ability to balance energy consumption of all sensor nodes and the WDM algorithm is the better one.

Finally, comparison is made between WDM algorithm and flooding algorithm with metric (C) to test energy efficiency, and the results are shown in Fig. 4. It is found that the flooding algorithm does pay much more energy costs to route one data packet and the WDM algorithm can gain better energy efficiency at all time. The average energy costs transferring one packet of the flooding algorithm is about 28.2 times of the WDM algorithm.

Hence, the results of the analyses above indicate that the WDM algorithm can gain quick data transmission, better evenness of dissipated network energy and energy efficiency to effectively extend network lifetime.

4 Conclusions

In this paper, advantages and deficiencies of existing multi-hop routing algorithms are first analyzed. The WDM algorithm, an energy-efficient weight-directed based multi-hop algorithm, is proposed and described. The results of a series of simulations of sensor network environments indicate that the WDM algorithm can have the ability to transfer data quickly, balance network energy consumption of all sensor nodes, improve energy efficiency, and accordingly extend system lifetime. This algorithm is well suitable for the continuous model of static distributed WSNs. In the future topological transformation for several nodes death will be conducted so as to improve the algorithm to suit for dynamic distributed WSNs.

Acknowledgments

This work is supported by the National High-Tech Research and Development Plan of China under Grant No. 2003AA1Z2080.

References

1. Weiser, M.: The Computer for the 21st Century. Sci. Amer., Sept. (1991)
2. Zengwei, Zheng and Zhaohui, Wu: A Survey on Pervasive Computing. Computer Science, Vol. 30, No. 4. Chongqing, China, Apr. (2003) 18-22, 29
3. Ogawa M., Tamura, T., Togawa, T.: Fully automated biosignal acquisition in daily routine through 1 month. International Conference on IEEE-EMBS, Hong Kong, Oct. (1998)
4. Mainwaring, A., Polastre, J., Szewczyk, R. and Culler, D.: Wireless Sensor Networks for Habitat Monitoring. ACM WSNA'02, Atlanta, Georgia, Sept. (2002)
5. Intanagonwiwat C., Govindan R., Estrin D.: Directed diffusion: a scalable and robust communication paradigm for sensor networks. Proceedings of the ACM MobiCom'00, Boston, MA, Aug. (2000)
6. Estrin D., Govindan R., Heidemann J., Kumar S.: Next Century Challenges: Scalable Coordination in Sensor Networks. Proceedings of the ACM MobiCom'99, Seattle, Washington, Aug. (1999)
7. Hedetniemi S., Liestman A.: A survey of gossiping and broadcasting in communication networks. Networks, Vol. 18, No. 4, winter (1988) 319-349
8. Tilak S., Abu-Ghazaleh N., Heinzelman W.: A Taxonomy of Wireless Micro-Sensor Network Models. ACM Mobile Computing and Communications Review (MC2R), Vol. 6, No. 2, Apr. (2002)
9. Zeng-wei Zheng, Zhao-hui Wu, Huai-zhong Lin: An Event-Driven Clustering Routing Algorithm for Wireless Sensor Networks. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), Sendai, Japan, Sept. (2004)
10. Sinha A., Chandrakasan A. P.: Energy Aware Software. Proceedings Of the 13th International Conference on VLSI Design, Calcutta, India, Jan. (2000)
11. Min R., Bhardwaj M., Cho S., Sinha A., et al.: An Architecture for a Power-Aware Distributed Microsensor Node. IEEE Workshop on Signal Processing Systems (SiPS '00) Design and Implementation, Lafayette, USA, Oct. (2000)

Forwarding Scheme Extension for Fast and Secure Handoff in Hierarchical MIPv6*

Hoseong Jeon[1], Jungmuk Lim[1], Hyunseung Choo[1], and Gyung-Leen Park[2]

[1] School of Information and Communication Engineering, Sungkyunkwan University, 440-746, Suwon, Korea
{liard, izeye, choo}@ece.skku.ac.kr
[2] Computer Science and Statistics Department, College of Natual Science, Cheju National University,
glpark@cheju.ac.kr

Abstract. Quality of service (QoS) and security in Mobile IP networks are becoming significant issues due to an increasing number of wireless devices [1]. For this reason, the Hierarchical Mobile IPv6 (HMIPv6) protocol [2] and the Authentication, Authorization, and Accounting (AAA) protocol [3] are proposed. However this protocol has inefficient authenticating and binding update procedures that limit its QoS. In this paper, we propose a forwarding scheme extension for fast and secure handoff that can reduce a handoff delay while maintaining a security level by a forwarding and session key exchange mechanism. The performance results show that the proposed mechanism reduces the handoff latency up to 10% and the handoff failure rate up to 25% compared to the previous mechanism.

1 Introduction

Based on mobility as the essential characteristic for mobile networks, the Mobile IP standard solution for use with the wireless Internet was developed by the Internet Engineering Task Force (IETF) [4]. However, Mobile IP does not extend well to highly mobile users. When a mobile node (MN) moves from one subnet to another one, it must send a location update to its home agent (HA) even though the MN does not communicate with others. These location updates incur the latency of messages traveling to the possibly distant home network [5]. Moreover, the term mobility implies higher security risks than static operation in fixed networks, since the traffic may at times take unexpected network paths with unknown or unpredictable security characteristics. Hence, there is a need to develop technologies that simultaneously enable IP security and mobility over wireless links.

* This work was supported in parts by Brain Korea 21 and the Ministry of Information and Communication in Republic of Korea. Dr. H. Choo is the corresponding author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 468–476, 2005.
© Springer-Verlag Berlin Heidelberg 2005

For this reason, the IETF suggests that the Hierarchical Mobile IPv6 (HMIPv6) and the Authentication, Authorization, and Accounting (AAA) protocol be employed. HMIPv6 adds hierarchy, built on MIPv6, which separates local from global mobility. In the HMIPv6, inter-handoff (global mobility) is managed by the MIPv6 protocols, while intra-handoff (local mobility) is managed locally.

In the basic AAA protocol, AAA server distributes the session keys to the MN and agents to guarantee security during data transmission. Yet, while an MN roams in foreign networks, a continuous exchange of control messages is required with the AAA server in the home network. Thus, the standard AAA handoff mechanism has inefficient authenticating procedures limiting its QoS. To resolve such problems, the forwarding scheme [6] and the session key exchange mechanism [7] are proposed.

The forwarding scheme is the proposed solution to the complications when the MN is required to send a binding update (BU) message to the HA during inter-handoff. In this scheme, the MN sends BU messages to a previous Mobility Anchor Point (MAP), subsequently the previous MAP forwards packets to a new MAP. The session key exchange mechanism essentially reuses the previously assigned session keys. This mechanism is important as it can drastically reduce the handoff delay. However, this mechanism requires that a trusted third party support the key exchange between the Access Routers (AR). For this reason, it uses only the intra-handoff within the same domain.

In this paper, we propose a modified session key exchange mechanism combined with a forwarding scheme. In Section 2, an overview of the HMIPv6 and AAA protocol is presented and the session key exchange mechanism and the forwarding scheme are given. Our proposed mechanism is discussed in Section 3. Performance evaluation for the proposed and previous methods follows in Section 4. Finally we conclude the paper in Section 5.

2 Preliminaries

In HMIPv6, global (between-site) mobility is managed by the MIPv6 protocol, while local (within-site) handoffs are managed locally. A new node in HMIPv6, termed the MAP serves as a local entity to aid in mobile handoffs. The MAP, which replaces MIPv4's foreign agent, can be located anywhere within a hierarchy of routers. In contrast to the foreign agent (FA), there is no requirement for a MAP to reside on each subnet. The MAP helps to decrease handoff-related latency since a local MAP can be updated faster than a HA of the MN.

Using MIPv6, a mobile node sends location updates to any node it corresponds with each time it changes its location, and at intermittent intervals otherwise. This involves a lot of signaling and processing, and requiring a lot of resources. Furthermore, although it is not necessary for external hosts to be updated when a mobile node moves locally, these updates occur for both inter and intra-handoffs. By separating inter and intra-handoff, HMIPv6 makes it possible to deal with either situation appropriately [2].

In this scheme, the MN moves around in a local domain based primarily on HMIPv6 as follows. The MN entering a MAP domain will receive a router advertisement message containing the information for one of several local ARs. It binds its current location with an address on the subnet of the MAP (RCoA). Acting as a local HA, the MAP will receive all packets on behalf of the MN and will encapsulate and forward them directly to the MN's current address. If the MN changes its current address within the local MAP domain (LCoA), it only needs to register the new address with the MAP. Hence, only in the beginning does the RCoA need to be registered with CNs and the HA. The RCoA remains constant as long as the MN moves around within the MAP domain. This makes the MN's mobility transparent to the CNs it is communicating with. Nevertheless, this protocol is restricted to apply only to the intra-handoff cases.

The Forwarding Scheme: The forwarding scheme improves the global mobility of HMIPv6. This scheme operates as follows. If the MN enters an initial regional network, and then the MAP_0 in its subnet will function as the MAP. When the MN enters a MAP_1 domain, it sends the BU message to the MAP_1, and the MAP_1 sends it back to the MAP_0. When the MAP_0 receives its message, it compares it to the MAP list and finds the MN's field. It then updates the current MAP address of the MN. After that, the MAP_0 relays the packet to the MAP_1 without the binding update through the HA. Fig. 1 shows the mechanism of the forwarding scheme [6].

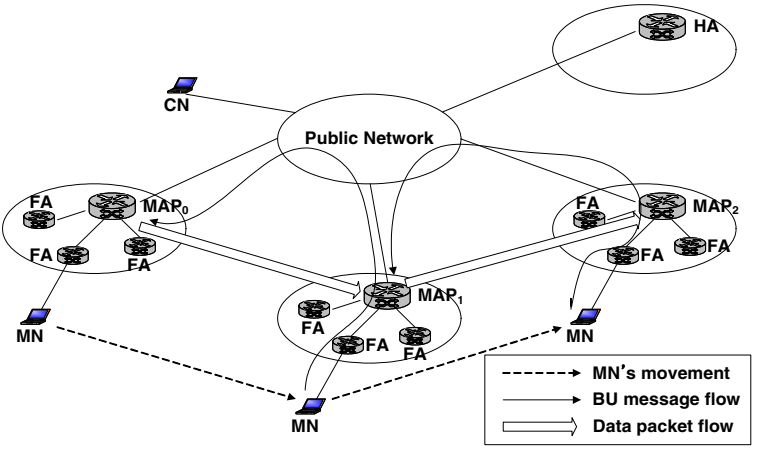


Fig. 1. The forwarding scheme

AAA Protocol: The IETF AAA Working Group has worked for several years to establish a general model for: Authentication, Authorization, and Accounting. AAA in a mobile environment is based on a set of clients and servers (AAAF and AAAH) located in the different domains. AAA protocol operates based on the security associations (SA_s: SA_1, SA_2, SA_3, and SA_4) as shown in Fig. 2.

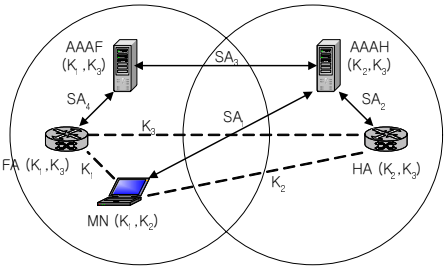


Fig. 2. AAA security associations

For the support regarding the secure communication, MN requires dynamic security associations. They are defined by sharing the session keys such as K_1, K_2, and K_3 between MN and HA, between HA and FA, and between FA and MN, respectively. Once the session keys have been established and propagated, the mobile devices can securely exchange data [8].

Session Key Exchange Mechanism: The Diffie-Hellman key agreement protocol depends on the discrete logarithm using two system parameters p and g. This scheme is based on a variant of Diffie-Hellman key agreement protocol instead of public key cryptography. Fig. 3 shows the session key exchange procedures. In fast operations, this scheme reuses the previously assigned session keys: the session keys for FA (S_{MN-FA} and S_{FA-HA}). To ensure the confidentiality and integrity of the session keys, it uses the encryption and decryption under a short lived secret key, $K_{oFA-nFA}$, between oFA and nFA. The key is dynamically shared between them and can be created by only two entities. However, there is a significant defect only applicable to intra-handoff [7].

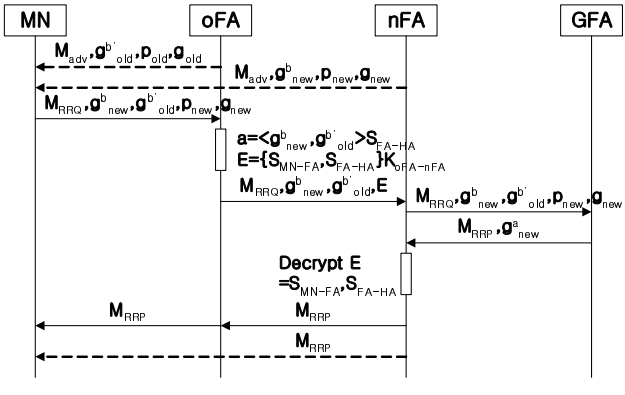


Fig. 3. Session key exchange mechanism

3 Proposed Mechanism

In this section, we describe the forwarding scheme for fast and secure handoff based on session key reuse. In this mechanism, the following assumptions are made:

- To prevent eavesdropping, all messages should be encrypted and exchanged in a secure process
- FAs related to the intra-handoff are trusted, that is, MAP authenticates them. Thus impersonating attack is not considered
- For the fast and secure inter-handoff, the AAA server can exchange session keys between FAs

The proposed mechanism improves the shortage of the previous authentication and binding update. The proposed scheme is divided into two parts according to the handoff type: 1) In the intra-handoff, our proposed scheme uses the session key reuse scheme by MAP and the micro-mobility management of HMIPv6. 2) In the inter-handoff, it uses the session key reuse scheme by AAA server and the forwarding mechanism.

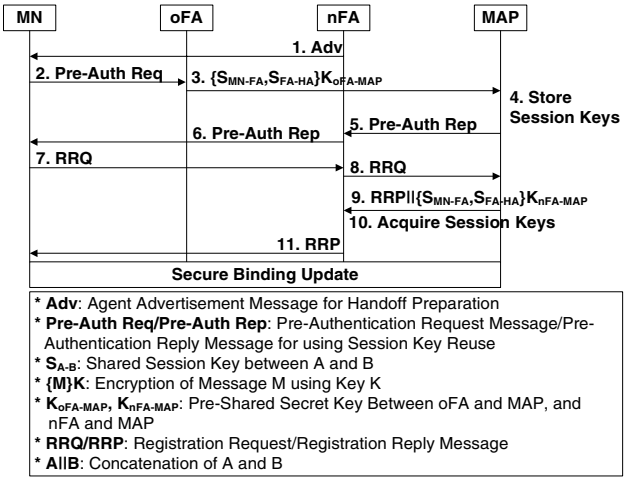


Fig. 4. The message procedure at intra-handoff

Fig. 4 shows the message procedure during intra-handoff. When the MN receives the agent advertisement message of nFA, it requests to the oFA for the reuse of the session key by sending Pre-Auth Req. After that, the oFA encrypts the session keys of itself by $K_{oFA-MAP}$ and then delivers them to the MAP. The MAP stores these session keys until it receives the registration request from the MN for the intra-handoff. If the MAP receives a RRQ message from the nFA, it sends a RRP message with $\{S_{MN-FA}, S_{FA-HA}\}K_{nFA-MAP}$. Finally, the nFA acquires these session keys and then sends a RRP message to the MN. Hence, the MN can send a binding update message in a secure fashion.

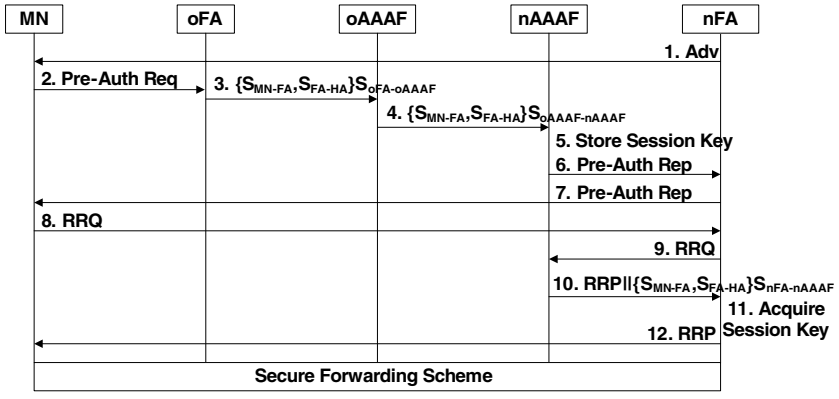


Fig. 5. The message procedure at inter-handoff

Fig. 5 shows the message procedure during inter-handoff. When the MN moves towards the region of nFA, it receives an agent advertisement message. Now the MN sends a pre-authentication request message. The oFA encrypts its session keys by the security association between the oFA and the oAAAF server, subsequently delivering it to the oAAAF server. The oAAAF server delivers it to the nAAAF server. The nAAAF server stores these session keys until it receives the registration request from the MN for the inter-handoff. If the nAAAF server receives a RRQ message from the nFA, it sends a RRP message with $\{S_{MN-FA}, S_{FA-HA}\}S_{nFA-nAAAF}$. Finally, the MN reduces the binding update time by using forwarding scheme while maintaining the security.

4 Performance Evaluation

4.1 Modeling

In order to evaluate the performance of our proposed algorithm, we make the following notations:

- $T_{MN-AR}/T_{AR-MAP}/T_{HA-MAP}/T_{MAP-MAP}/T_{MAP-AAA}/T_{AAA-AAA}$: The transmisson time between the MN and AR/the AR and MAP/the HA and MAP/MAPs/the MAP and AAA server/AAA servers, respectively.
- $P_{AR}/P_{HA}/P_{MAP}/P_{AAA}$: The processing time at the AR/the MAP/the HA/ the AAA server, respectively.
- T_H/T_R: The registration time of the home registration time/the regional registration time, respectively.
- T_M: The time to establish a link between MAPs
- $A_H/A_R/A_M$: The authentication time based on the basic AAA protocol/the session key reuse scheme by MAP/the session key reuse scheme by AAA server, respectively.

We calculate times required for the performance evaluation using the following equations as above notations. First of all, the HMIPv6 binding update time (BU) is represented as:

$$BU_{Intra}^{HMIPv6} = 2T_{MN-AR} + 2T_{AR-MAP} + 2P_{AR} + 2P_{MAP} \quad (1)$$

$$BU_{Inter}^{HMIPv6} = 2T_{MN_AR} + 2T_{AR-MAP} + 2T_{MAP-HA} + 2P_{AR} + 2P_{MAP} + P_{HA} \quad (2)$$

In the proposed scheme, we assume that the MN moves between the MAPs, and thus the binding update time is calculated as:

$$BU^{Proposed} = 2T_{MN-AR} + 2T_{AR-MAP} + 2T_{MAP-MAP} + 2P_{AR} + 3P_{MAP} \quad (3)$$

The total authentication time (AT) in the standard AAA protocol is acquired as follows:

$$\begin{aligned} AT^{Std} = {} & 2T_{MN-AR} + 2T_{AR-MAP} + 2T_{MAP-AAA} + 2T_{AAA-AAA} + \\ & AS + 2T_{MAP-AAA} + 2T_{MAP-HA} + 2P_{AR} + 4P_{MAP} + 2P_{HA} \end{aligned} \quad (4)$$

Finally, the total authentication time in the proposed scheme is calculated as shown below.

$$AT_{Intra}^{Proposed} = 2T_{MN-AR} + 2T_{AR-MAP} + 2P_{AR} + 2P_{MAP} \quad (5)$$

$$\begin{aligned} AT_{Inter}^{Proposed} = {} & 2T_{MN-AR} + 2T_{AR-MAP} + 2T_{MAP-AAA} \\ & + 2T_{AAA-AAA} + 2P_{AR} + 4P_{MAP} + 2P_{AAA} \end{aligned} \quad (6)$$

The probability (P_f) in which the MN leaves the boundary cell before the required time T_{req} is represented as $Prob(T < T_{req})$, where we assume T is exponentially distributed. Thus, the handoff failure rate as follows: $P_f = 1 - exp(-\lambda \cdot T_{req})$. λ is the arrival rate of MN into the boundary cell and its movement direction is uniformly distributed on the interval [0, 2π). Thus λ is calculated by the equation $\lambda = V \cdot L \ / \ \pi \cdot S$ [10]. Here V is the velocity for MN and L is the length of the boundary and S is the area of boundary. Hence we obtain the handoff failure rate by T_{req} and λ.

4.2 Analytical Results

Using above equations and the system parameters in Table 1 [5, 9, 10], we compute the cumulative handoff delay and the handoff failure rate. As shown in Fig. 6, our proposed scheme does not limit the number of forwardings as it always shows the better performance in the cumulative handoff latency. Consequently, our proposed scheme is limited by the freshness of the session key.

We perform an analysis of the handoff procedure to obtain the handoff failure rate according to each handoff mechanism. The handoff failure rate is influenced by few factors: the velocity of MN and the radius of a cell. Fig. 7 shows the result of the handoff failure rate. The proposed scheme consistently shows the better handoff failure rate in comparison with previous mechanisms.

Table 1. System parameters

Bit rates		**Processing time**	
Wire/Wireless	100/2 *Mbps*	MN/AR/MAP/AAA	0.5 *msec*
Propagation time		3DES	0.5 *msec*
Wire/Wireless	0.5/2 *msec*	MAC (Message Authentication Code)	0.5 *msec*
Data size		AS (Authentication time in server)	6.0 *msec*
Message size	256 *bytes*		

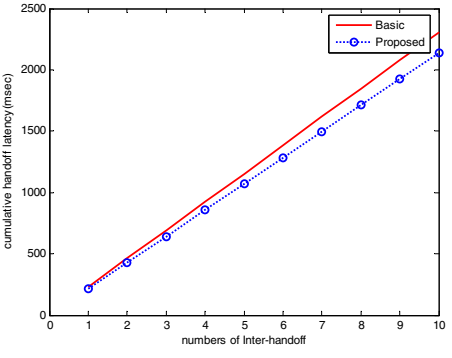


Fig. 6. The cumulative handoff latecny

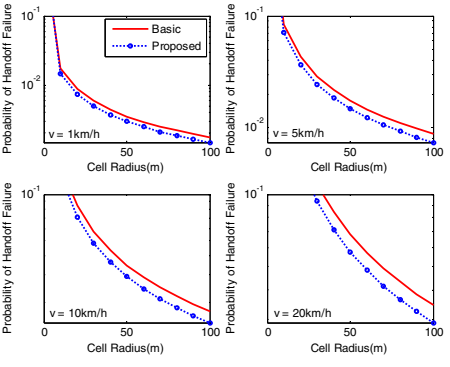


Fig. 7. The handoff failure rate

5 Conclusions

In this paper, we have proposed the forwarding scheme extension for fast and secure handoff employing a forwarding scheme and session key exchange mechanism in order to provide reduced handoff latency while maintaining the previous mechanism's security level. The performance comparison results show that the proposed mechanism is superior to the previous ones in terms of handoff latency while maintaining the security level. We are currently conducting an analysis of the threshold of the session key freshness.

References

1. C. Perkins, "IP Mobility Support," IETF RFC 2002.
2. H. Soliman, "Hierarchical Mobile IPv6 mobility management (HMIPv6)", IETF, October 2004.
3. C. Perkins, "Mobile IP Joins Forces with AAA," IEEE Personal Communications, vol. 7, no. 4, pp. 59–61, August 2000.
4. D. Johnson, "Mobility Support in IPv6", RFC 3775, IETF, June, 2004.
5. J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. debruijn, C.de Laat, M. Holdrege, and D. Spence, "AAA Authorization Application Examples," IETF RFC 2905.

6. D. Choi, H. Choo, J. Park, "Cost Effective Location Management Scheme Based on Hierarchical Mobile IPv6," Springer-Verlag Lecture Notes in Computer Science, vol. 2668, pp. 144–154, May, 2003.
7. H. Kim, D. Choi, and D. Kim, "Secure Session Key Exchange for Mobile IP Low Latency Handoffs," Springer-Verlag Lecture Notes in Computer Science, vol. 2668, pp. 230–238, January 2003.
8. C. de Laat, "Generic AAA Architecture," RFC 2903, IETF, August, 2000.
9. H. Jeon, H. Choo, and J. Oh, "IDentification Key Based AAA Mechanism in Mobile IP Networks," ICCSA 2004 vol. 1, pp. 765–775, May 2004.
10. J. McNair, I.F. Akyildiz, and M.D Bender, "An inter-system handoff technique for the IMT-2000 system," INFOCOM 2000, vol. 1, pp. 203–216, March 2000.

Back-Up Chord: Chord Ring Recovery Protocol for P2P File Sharing over MANETs*

Hong-Jong Jeong†, Dongkyun Kim†, Jeomki Song†,
Byung-yeub Kim‡, and Jeong-Su Park‡

† Department of Computer Engineering,
Kyungpook National University, Daegu, Korea
{hjjeong, jksong}@monet.knu.ac.kr, dongkyun@knu.ac.kr
† Electronics and Telecommunications Research Institute, Daejoen, Korea
{skylane, pjs}@etri.re.kr

Abstract. Due to a common nature of MANET (Mobile Ad Hoc Networks) and P2P (Peer-to-peer) applications in that they lack a fixed infrastructure, a P2P application can be a killer application over MANET. To save network bandwidth and avoid a point of failure of a directory server, structured P2P systems using DHT (Distributed Hashing Table) like Chord are more suitable for MANET. However, since MANET allows nodes to depart from network, P2P file sharing applications based on Chord lookup protocol should address how to recovery the keys stored at the departed node. In this paper, we propose BU-Chord (Back-Up Chord) in order to detect and recover the departure of nodes by creating and storing a back-up file information in distributed manner. Simulation study proves that our BU-Chord shows off better performance than the existing Chord especially at high departure rate of nodes.

1 Introduction

Recently, research interest in MANET (Mobile Ad Hoc Networks) [1] has increased because of the proliferation of small, inexpensive, portable, mobile personal computing devices. MANET is a wireless network where all nomadic nodes are able to communicate each other through packet forwarding services of intermediate nodes. Besides, from the application's perspective, a P2P (Peer-to-peer) model is prevalent to enable a direct communication between nodes in the network [2]. Many file sharing applications such as Napster [3]and Gnutella [4] rely on this P2P concept. Due to a common nature that MANET and P2P applications both assume a lack of fixed infrastructure, a P2P application can be a killer application over MANET [5]. In centralized systems like Napster, a centralized directory server has the information on who has which files. However, the centralized approach is not suitable for MANET because the server can easily move out of the MANET due to node mobility.

* This work was supported by Electronics and Telecommunications Research Institute (ETRI). The corresponding author is Dongkyun Kim.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 477–484, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Fully distributed systems like Gnutella do not depend on the existence of a centralized server. A query message for a file search is flooded into the network. Such a distributed approach is also not suitable for MANET because the query flooding produces much traffic in the network with scarce resource. In order to avoid the query flooding, structured P2P systems using the DHT (Distributed Hash Table) mechanism such as Chord [6] were developed. Particularly, Chord distributes files and their references into the network through the DHT technique. Chord forms an overlay network, where each Chord node needs "routing" information about only a few other nodes, as well as "file information" shared among nodes and used in order to know who has requested files.

However, since MANET allows nodes to depart from network, it is difficult to apply the Chord to MANET because it cannot recover the file information. In this paper, we therefore propose BU-Chord (Back-Up Chord) protocol in order to detect and recover the departure of nodes, efficiently. BU-Chord creates and stores a back-up file information in distributed manner. Although this paper applies the BU-Chord to MANET, it can be used in any network environment, where failure of nodes occurs frequently, because the failure of nodes is equivalent to the departure of nodes out of networks.

The rest of this paper is organized as follows. In Section 2, the basic Chord is introduced in short. In Section 3, our BU-Chord protocol is described in detail. We perform the performance evaluation in Section 4, which is followed by concluding remarks in Section 5.

2 Chord: A Scalable P2P Lookup Protocol

Like most of other structured P2P protocols using the DHT technique, Chord defined assignment and lookup mechanisms, where a key is used as a name of a shared file. Each Chord node and key obtain their unique m-bit identifiers by using a base hash function such as SHA-1 [7]. A node's identifier is determined by hashing the node's IP address, while a key identifier is produced by hashing the key. Using the node identifier, the Chord creates an identifier space (from 0 to 2^m - 1), which is called "Chord ring" and distributed in the network.

Chord utilizes a consistent hashing to assign keys to Chord nodes. Key k is assigned to the first node whose identifier is equal to or follows (the identifier of) k in the identifier space. The first node is called the successor of key k, denoted by $successor(k)$. In order for each Chord node to perform a lookup for finding a successor of key k, it should manage its successor, predecessor and finger table. A successor and a predecessor are the next and the previous node in the identifier space, respectively. Finger table consists of m entries. The i^{th} entry in a finger table of node n is $successor(n + 2^{i-1})$, where $1 \leq i \leq m$. In the steady state, in an N-node system, each node maintains information about only O(log N) other nodes, and resolves all lookups through O(log N) messages to other nodes.

Figure 1 shows a Chord ring using m-bit identifier (here, $m = 6$). The ring consists of 10 nodes and has 5 keys assigned. In order for Chord to find a suc-

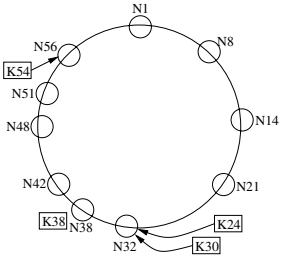


Fig. 1. An example of Chord ring constructed

cessor which will manage a key, it uses a consistent hashing. Therefore, keys are distributed in the Chord ring, which results in providing a degree of natural load balance.

3 BU-Chord: Back-Up Chord

3.1 Motivation

Chord's lookup algorithm enables a wireless network bandwidth to be saved by avoiding the query flooding due to the DHT mechanism. For the purpose of sharing files among Chord nodes, each Chord node should assign a key k (i.e. a name of a shared file) to $successor(k)$ by using the consistent hashing. The searching for a file can be completed by performing a lookup for key k corresponding to the file, just as k is assigned to $successor(k)$. However, since MANET allows nodes to depart from the network at any time, a departure of node causes the network to lose the keys stored at the node. Therefore, any node cannot search for the keys.

To address loss of the keys in the network, other Chord nodes should have the replication of the keys in advance. In this paper, we propose our BU-Chord (Back-Up Chord) to replicate the keys stored at each node into other nodes and recover the keys stored at departed nodes.

In other research work, an effort to improve the reliability of Chord's lookup process over MANET was made [8]. However, for the purpose of developing a P2P file sharing application based on the Chord lookup protocol, the problem should be addressed that the departure of a node causes the keys stored at the node to be lost in the network. Therefore, our BU-Chord can be utilized as their complementary protocol.

3.2 Description of Proposed Approach

In this section, we describe our BU-Chord (Back-Up Chord) protocol to detect a node departure and allow a back-up node to recover the keys stored at the departed node. BU-Chord utilizes a concept of back-up successor and predecessor to replicate the keys stored at a node. Each node performing BU-Chord

protocol is assigned m-bit back-up identifier as well as m-bit identifier by using a hash function such as SHA-1. According to the existing Chord protocol, a node obtains a m-bit identifier by hashing its own IP address. In BU-Chord protocol, each node produces an additional m-bit back-up identifier by hashing its derived m-bit identifier again. A successor of the derived back-up identifier (called back-up successor) is determined by using the same consistent hashing that the existing Chord protocol uses. In BU-Chord protocol, each node requires its back-up successor to replicate its keys and the information on its successor and predecessors of a BU-Chord ring. The back-up successor regards the requesting node as its back-up predecessor and performs a periodical procedure to check if its back-up predecessor is alive in the network through exchange of BEACON/BEACON-ACK message[1].

With the absence of BEACON-ACK, a back-up successor considers that its back-up predecessor departed from the network and performs a recovery process. Since the back-up successor knows who the successor and predecessor of the departed node (say, DN) are, it forces the successor (say, SN) of the DN to update its predecessor with the predecessor of the DN. This procedure is used to just recover a broken Chord ring. As a next step, the keys stored at the DN should be moved to other node. In BU-Chord protocol, since the back-up successor of the DN knows the keys, it can move the keys into the SN, because the SN becomes a successor of the keys. On the other hand, when the back-up predecessor cannot receive its BEACON-ACK from its back-up successor, it is enough to figure out a new back-up successor and replicate its keys and the information on its successor and predecessors of a Chord ring there.

Figure 2 illustrates a recovery operation of our BU-Chord protocol. Assume that N25 has departed and its back-up identifier is B53 and its back-up successor is N57. The predecessor and successor of N25 are N19 and N36, respectively. Through a periodic procedure of N57, N25 is recognized as departed. N57 forces the successor of N25 (here, N36) to update its predecessor with N19 for recovering the Chord ring. Then, N57 moves the information on K22 and K23 into N36.

3.3 Departures of Multiple Nodes

In all cases except a case where both successor of a departed node and its back-up successor disappear at the same time, BU-Chord described in Section 3.2 recovers the keys stored at the departed node. Simultaneous departures of multiple nodes can be broken down into three cases: (i) a departed node (DN) and its successor (SN) move out, (ii) a departed node (DN) and its back-up successor (BN) move out, (iii) a departed node (DN), its successor (SN) and back-up successor (BN) move out.

To address the departures of multiple nodes, some additional mechanism should be executed at back-up successors as described below. The first case can

[1] Proactive MANET routing protocols standardized in IETF (Internet Engineering Task Force) such as OLSR and TBRPF (see [1]) are suitable for reducing the overhead expended to establish a route for exchanging the messages.

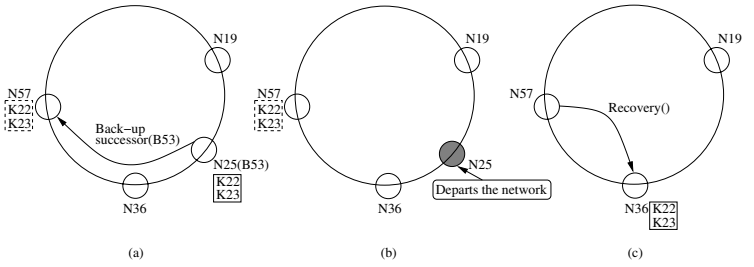


Fig. 2. An Example of BU-Chord Recovery. (a) N57 is a back-up successor of N25. N57 replicates the keys (i.e. K22 and K23) stored at N25. (b) N25 departs the network. (c) N57 moves the keys (i.e. K22 and K23) to N36

be resolved by allowing a back-up successor of an SN to have additional recovery procedure. The BN of a DN attempts to recover the keys into the SN of the DN. The BN checks if the SN works through exchange of BEACON/BEACON-ACK. However, the BEACON/BEACON-ACK exchanging will fail because the SN moved out from the network. Therefore, after the BN considers that both DN and SN moved out, it carries out this following recovery procedure. After BN obtains a back-up identifier of the SN by hashing the node identifier of the SN, it lookups the back-up successor of the SN, i.e. successor(back-up identifier of the SN). Thereafter, the keys stored at the DN are moved into a back-up successor of the SN. Then, the back-up successor of the SN also moves the keys into a successor of the SN.

Figure 3 shows an example of the first case. Suppose that in Figure 2, a successor of N25, i.e. N36 also moved out. The back-up identifier of N36 is B47 and its back-up successor is N49. The predecessor and successor of N36 are N25 and N39, respectively. N57, which is a back-up successor of N25, becomes aware of the departure of N25 and sends a recovery request to N36, the successor of N25. However, the trial will fail because N36 also moved out. Therefore, N57 should try to send a recovery request to a back-up successor of N36, i.e. N49. However, N57 does not know who the back-up successor of the disappeared N36 is. According to our BU-Chord's mechanism through which a back-up successor is determined, the back-up successor of N36 is decided and a recovery request for the keys stored at N25 (i.e. K22 and K23) can be issued. N57 obtains the back-up identifier of N36 (i.e. 47) by hashing the node identifier of N36 (i.e. 36). Therefore, the successor (i.e. N49) of the back-up identifier can be found. Thereafter, N57 provides the back-up successor of N36 (i.e. N49) with the keys stored at N25 (i.e. K22 and K23) and the information on the predecessor of N25 (i.e. N19). The recovery is completed after N49 provides the successor of N36 (i.e. N39) with the keys stored at N25 and the keys stored at N36 (i.e. K35 and K36).

The second case can be resolved by allowing a back-up successor of a BN to perform an additional recovery procedure. After the back-up successor of a BN can recognize that the BN moved out, the keys stored at the BN can be recovered

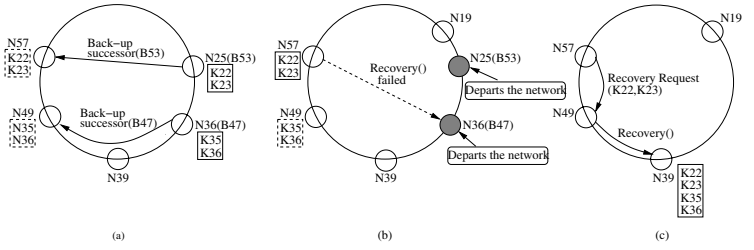


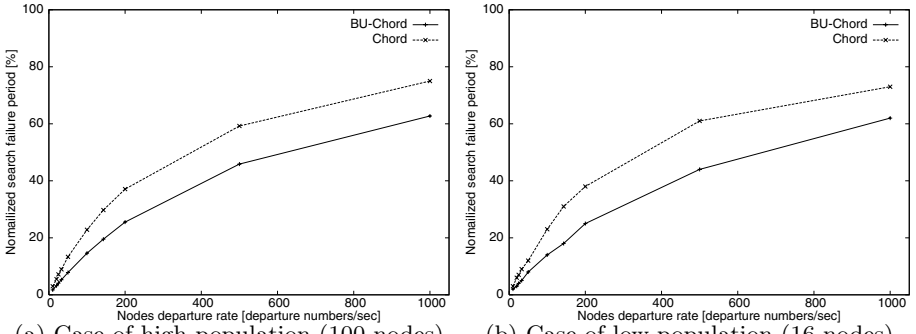
Fig. 3. Both departed node and its successor moved out. (a) N49 and N57 are the back-up successors of N36 and N25, respectively. (b) N25 and N36 depart the network at the same time. (c) N57 provides N49 with K22 and K23. Thereafter, N49 provides N39 with K22, K23, K35 and K36

by moving the keys stored at the BN into a successor of the BN. Thereafter, an additional procedure to check if an actual departure of multiple nodes occurred. The back-up successor of the departed BN executes a recovery of the keys stored at the BN and then, checks if a back-up predecessor of the BN (i.e. the DN) works through exchange of BEACON and BEACON-ACK. Therefore, if two nodes, the BN and its back-up predecessor moved out together, the back-up successor of the BN moves the keys already replicated at the BN, which are originally stored at the DN, into the successor of the DN.

In the final case, the keys stored at nodes, the BN and SN are recoverable through the recovery procedure executed by the BN, SN and each back-up successor of the BN and SN.

4 Performance Evaluation

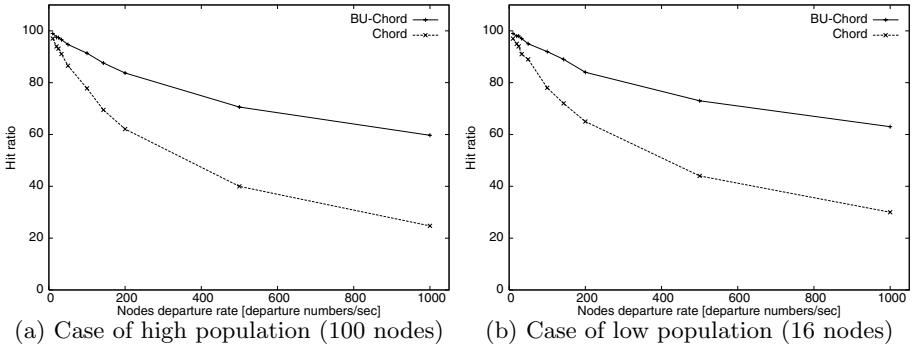
The existing Chord does not define how to recover keys stored at each Chord node. Therefore, we assumed that in Chord, if the keys stored at a departed node are lost in the network, the nodes having the files corresponding to the keys assign them to a new successor of each key. We investigated performances under two kinds of MANETs: (i) high population and (ii) low population. When deploying our BU-Chord over MANET, we assumed that proactive MANET routing protocols are used to reduce the overhead expended to establish a route before moving keys and exchanging BEACON/BEACON-ACK message. For configuring the population of nodes, 100 nodes and 16 nodes are positioned in a grid-style network topology, respectively. We compared BU-Chord with Chord by varying the departure rate of nodes. In our simulation, each node performs search trial periodically by using a normal distribution, where the average interval is 1 second. Each key value which each node would like to find is randomly generated. When a query message succeeds in reaching a successor having the key, it is regarded as search success. Otherwise, we regards the other case as search failure. Therefore, if the keys are not recovered due to node departure, any search for the key will fail.



(a) Case of high population (100 nodes) (b) Case of low population (16 nodes)

Fig. 4. Comparison of normalized search failure period

First, we measured a normalized search failure period (NSFP) according to departure rate of nodes. Since the keys stored at a departed node are recovered by a back-up successor of the departed node, we define NSFP as a portion of the average time during which we cannot perform search success for each key during the total simulation time. As shown in Figure 4, as the departure rate increases, NSFP also increases. Irrespective of departure rate and population of nodes, our BU-Chord shows better NSFP than Chord because BU-Chord allows a back-up successor of a departed node to quickly detect its departure and recover the keys stored at the departed node.



(a) Case of high population (100 nodes) (b) Case of low population (16 nodes)

Fig. 5. Comparison of hit ratio

Second, we investigated hit ratio according to departure rate of nodes, which is defined as a ratio of the number of search success to the total number of search trials. We observed that as the departure rate increases, the hit ratio is decreasing. Obviously, BU-Chord performs better than Chord without regard

to node population and departure rate (see Figure 5). In particular, BU-Chord shows off its performance improvement at high departure rate of node.

5 Conclusion

Using Chord which is a typical structured P2P file sharing application, we can save the network bandwidth by avoiding query flooding in MANET, because keys are distributed in the network. However, when the existing Chord is applied to MANET, departure of nodes forces the keys stored at the nodes to be lost in the network, which result in inability of searching for the keys. However, in our proposed BU-Chord (Back-Up Chord), a back-up successor of each node has the replication of each node's keys and the information on neighboring nodes. Due to the replication technique, the back-up successor detects the departure of node and recovers the keys stored at the departed node. In particular, at high departure rate of nodes in MANET, BU-Chord showed off better performance than Chord irrespective of population of nodes. Although the BU-Chord is applied to MANET in this paper, it can be used in any network environment, where failure of nodes occurs frequently, because the failure of nodes is equivalent to the departure of nodes out of networks.

References

1. Internet Engineering Task Force, "Manet working group charter," http://www.ietf.org/html.charters/manet-charter.html.
2. Gang Ding and Bharat Bhargava, "Peer-to-peer File-sharing over Mobile Ad hoc Networks," IEEE PERCOMW 2004, Mar.2004.
3. Napster, http://www.napster.com.
4. The Gnutella Protocol Specification v0.4.
5. L.B.e. oliveira, I.G. Siqueira and A.A..F Loureiro, "Evaluation of Ad-Hoc Routing Protocol under a Peer-to-Peer Application," IEEE WCNC 2003, Mar. 2003.
6. I. Stoica, R. Morris, D.L. Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Transactions on Networking, Vol. 11, No. 1, Feb. 2003.
7. "Secure Hash Standard," U.S. Dept. Commerce/NIST, National Technical Information Service, Springfield, VA, FIPS 1801-1, Apr. 1995.
8. S.Y. Lee, L. Quan, K.G. Lee, S.Y. Lee, and J.W. Jang, "Trade-off between Message Overhead and Reliability for Peer-to-Peer Search over Mobile Ad Hoc Networks," ISPC COMM 2004, Aug., 2004.

PATM: Priority-Based Adaptive Topology Management for Efficient Routing in Ad Hoc Networks

Haixia Tan, Weilin Zeng, and Lichun Bao

Donald Bren School of Information and Computer Sciences,
University of California, Irvine, Irvine, CA 92697
{htan, wzeng, lbao}@ics.uci.edu

Abstract. We propose a distributed and adaptive topology management algorithm, called PATM (Priority-based Adaptive Topology Management), that constructs and maintains a connected backbone topology based on a minimal dominating set of the network. PATM provides a succinct presentation of the network topology to routing protocols, and therefore reduces the control overhead in routing updates. Two optimizations are proposed to further reduce the topological information exchanges among the nodes by piggybacking topology updates in packets transmitted by each node, and adaptively adjusting the topology update intervals. The efficiency of the algorithm is validated by simulations based on DSR (Dynamic Source Routing) protocol. The simulation results demonstrate that PATM not only significantly reduces the routing control overhead, but also substantially improves the network data forwarding performance.

1 Introduction

Different from most cellular networks which are supported by a fixed, wired infrastructure, and scheduled by the central base stations, ad hoc networks are self-organizing, self-configuring wireless networks. Topology management has been proposed as an effective and efficient approach to performing some control functionalities in ad hoc networks. The main task of topology management is to select an appropriate subset of the original topological network graph. The backbone constructions are usually based on hierarchical clustering, which consists of selecting a set of clusterheads that covers every other node, and are connected with each other by means of gateways.

Different clustering algorithms propose different criteria in selecting clusterheads and gateways [3][7][11][6][8]. SPAN [12] adaptively elects coordinators according to the remaining energy and the number of pairs of neighbors a node can connect. GAF [1] subdivides a sensor network into small grids, such that only one node in each grid is active at each point of time. PILOT [2] proposed to use a set of mobile nodes in the sensor network to bridge failing connections. In ASCENT [5], a node decides to join the backbone based on the number of neighbors and the data message loss probability at the node. STEM [4] also saves power by turning off a node's radio. It adaptively puts nodes to sleep and to wake up nodes only when they need to forward data.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 485 – 492, 2005.
© Springer-Verlag Berlin Heidelberg 2005

TMPO [6] proposes to construct and maintain a network backbone based on MDS (*Minimal Dominating Set*) and CDS (*Connected Dominating Set*) using only two-hop neighbor information. CEC [7] is another distributed, proactive clustering algorithm. In CEC, Clusterhead election is based on the lifetime of each node. The gateways are elected according to the node's degree. CEC is an improvement of Geographic Adaptive Fidelity (GAF [1]), which relies on location information.

Unlike TMPO, CEC, and WCA [8], On-Demand Cluster Formation (ODCF [11]) is a reactive, on-demand clustering algorithm. Adaptive Clustering (AC [3]) proposed to use clustering for different tasks, such as spatial reuse of bandwidth, Quality of Service (QoS) provisioning by resource allocation within clusters.

We propose a novel hierarchical clustering algorithm, Priority-based Adaptive Topology Management (PATM), which is adaptive to the dynamic changes of topology, bandwidth resource availability and traffic loads. We show that cluster-based control mechanisms can significantly reduce the overhead in routing, while improving data forwarding services. In comparison with other clustering algorithms (such as GAF [1], PILOT [2], and TMPO [6], etc.), PATM distinguishes itself by combining these features: (1) It does not require node position information or synchronization among nodes. (2) It does not need centralized control over the ad hoc network. Every node makes decisions based on its local information. (3) It proactively maintains a connected backbone, but without exchanging control messages periodically. Furthermore, topology updates are dramatically reduced using two optimizations. The first one is to piggyback the small control messages to the ongoing traffic. The second is to adapt the topology update intervals based on the network mobility.

The rest of the paper is organized as follows. Section 2 describes PATM algorithm. Section 3 presents extensive simulation results by running DSR with and without topology management using PATM. Section 4 summarizes this paper.

2 PATM

2.1 Priority Computation

PATM is a distributed clustering algorithm for constructing the connected dominating set of the network by comparing the priorities of two-hop neighbors. The priority of a node, say i, is a function of the node's ID, current time slot number t, the remaining energy Ei and the moving speed of the node Si using the following formula:

$$Pi = h(i, t, Ei, Si)$$

which gives low priority at high speed or low energy situations.

2.2 Information Exchange

PATM requires nodes in an ad hoc network directly exchange priority information of themselves and that of their one-hop neighbors. This change in the information exchange allows the nodes to adapt the interval of their priority computations according to the network traffic and mobility conditions, instead of carrying out the priority computation by other nodes periodically. When network traffic load or mobility varies at different parts of the network, nodes can be more active or passive in forming

the backbone of the network. For example, when a region of the network carries very light traffic, PATM can increase the interval of priority updates, causing less control overhead, and more energy savings.

In addition to exchanging the priority information of the nodes, the clusterhead status of a node and its one-hop neighbors are also exchanged by broadcasts.

i	*Pi*	*i.type*
j	*Pj*	*j.type*
k	*Pk*	*k.type*

Fig. 1. Information in a PATM Update Packet

i	*Pi*	*i.type*
j	*Pj*	*j.type*
k	*Pk*	*k.type*
N_j^1	-	-
N_k^1	-	-

Fig. 2. Node *i*'s Neighbor Table

As an example where node *i* has two one-hop neighbors *j* and *k*, that is, $N_i^1 = \{j, k\}$, node *i* broadcasts a packet with the information shown in Fig. 1. Similarly, node *j* broadcasts the information about itself and N_j^1. So does node *k*.

According to the neighbor information exchanged, every node acquires and maintains a neighbor table that stores the information about its one-hop and two-hop neighbors including the priority and the type of each node. Following the same example given in Fig. 1, the content of node *i*'s neighbor table is shown in Fig. 2.

The last two rows in Fig. 3 is an abbreviation of all the one-hop neighbors of node *j* and *k*, which is dependent on the concrete topology of the network. The corresponding attributes of the members in the last two rows are omitted.

2.3 Clusterhead Election

Without loss of generality, we describe PATM clusterhead election algorithm from node *i*'s point of view. First, node *i* initializes its own type as *host*. Then it decides to become a *clusterhead* if either one of the following criteria are satisfied.

(1) Node *i* has the highest priority in its one-hop neighborhood.
(2) Node *i* has the highest priority in the one-hop neighborhood of one of its one-hop neighbors.

2.4 Doorway and Gateway Election

After the MDS is formed, the CDS is constructed in two steps.

(1) If two clusterheads in the MDS are separated by three hops and there are no other clusterheads between them, a node with the highest priority on the shortest paths between them is elected as a *doorway*, and becomes a member of the CDS.
(2) If two clusterheads or one clusterhead and one doorway are only two hops away, and there are no other clusterheads between them, one of the nodes between them

with the highest priority becomes a *gateway* connecting the clusterhead to another clusterhead or the doorway to the clusterhead, and becomes a member of the CDS.

As an example, Fig. 3 (a) shows the topology of an ad hoc network. Fig. 3 (b) shows a possible result of applying topology management and forming the CDS.

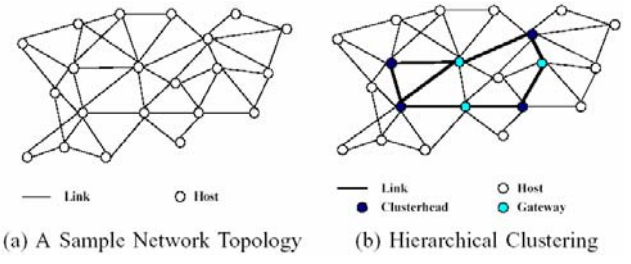


Fig. 3. Topology Management

2.5 Piggybacking Optimization

In PATM, nodes in networks have to exchange routing control information to maintain the connectivity of the networks in mobile environments. Therefore, instead of sending out topology management update packets alone, we apply the piggybacking mechanism to the ongoing topology update packets whenever necessary. The outgoing packets are those sent by the network layer, which could be regular data packets, or routing control messages, such as Route Request (RREQ) and Route Reply (RREP) in DSR.

2.6 Adaptation to Network Mobility

The adaptation of PATM to network mobility is based on a key observation that the interval between re-computing the node priorities and sending topology updates is critical for the network performance. If the interval is too short, the control packet overhead will increase dramatically, and if the interval is too large, the CDS in PATM may not be able to catch up with the topology change. In PATM, the interval of re-computing the node priorities and updating neighbor information varies at different nodes. Each node determines its own interval value.

The frequency of one-hop neighbor changes during the current update interval is taken as an indicator of the relative speed in deciding the next update interval in PATM. As shown before, every node maintains a neighbor table in PATM. The number of one-hop neighbor changes during the current interval *Ti* is used to update the next interval value *Ti* for topology updates. In addition, if node *i* has not received any packets from a one-hop neighbor for a certain period of time, this one-hop neighbor and its associated two-hop neighbors will be deleted from the table.

Fig. 4-6 describe the essential functions of PATM using C-style pseudo-codes. Fig. 4 provides the initialization of various variables in PATM. Fig. 5 specifies the callback function after each update interval to adjust the next update interval. For conven-

ience, the factors adjusting the interval value are given in the algorithm, which performs well in the simulations. However, they are tunable parameters. Fig. 6 provides the condition for piggybacking topology updates to the outgoing packets.

```
Init(i) {
1  piggybacked = FALSE;
2  oneHopChangeNum = 0;
3  Ti = 10;
4  Schedule(Callback, Oi);
}
```

Fig. 4. Initialization in PATM

```
Callback(i) {
    // Re-compute priority
1   t = Current_time();
2   Pi = h(i, t, Ei, Si);
3   if (!piggybacked)
4     Propagate_topology();
5   piggybacked = FALSE;
6   Check_one_hop_nbr();
7   if(oneHopChangeNum <= 1)
8     Ti *= 2;
9   else if (oneHopChangeNum <= 4)
10    Ti *= 1.5;
11  else if (oneHopChangeNum >= 8
      && oneHopChangeNum <= 15)
12    Ti *= 0.75;
13  else if (oneHopChangeNum > 15)
14    Ti = 30; //Ti is 30 seconds.
15  oneHopChangeNum = 0;
    // Schedule the next callback.
16  Schedule(Callback, Ti);
}
```

Fig. 5. PATM Function for Maintenance

```
Piggyback(i) {
1  if (Current_time()-piggyback_time
     > Ti-2) {
2    Piggyback_topology( );
3    piggyback_time = Current_time();
4    piggybacked = TRUE;
5  }
}
```

Fig. 6. PATM Function for Piggyback

Before any outgoing packet is sent down to the network interface, function `Piggyback()` is invoked to see if there is a topology update packet ready to piggyback. The variable `piggyback_time` records the time when a piggyback happens (`Piggyback()` line 3). To prevent the piggyback procedure from happening too frequently, piggybacking happens only if the time difference between the current time and `piggyback_time` is greater than a threshold (`Piggyback` line 1). In addition, the same topology information does not have to be piggybacked in every outgoing packet because the transmission delay increases as the size of each packet increases (`Piggyback`() line 4 and `Callback`() line 3-4).

If the topology information has not gotten an opportunity to be piggybacked during the period, node *i* will broadcast it in a separate control packet (`Callback` line 3-5) so as to guarantee the topology information is broadcast at least once in a period.

In Fig. 5, function `Check_one_hop_nbr()` is not specified, but is used to check the validation of every element in N_i^1. Variable `oneHopChangeNum` is used to record the number of one-hop neighborhood changes during the current update period. The variable is set to 0 in the initialization (`Init()` line 3) and at the beginning of each period (`Callback` line 15). Thereafter, every time when a one-hop neighbor is inserted or deleted from the neighbor table, the variable will be increased by 1. At the end of the period, the value of `oneHopChangeNum` is checked as the relative speed to adjust the length of the next period (`Callback` line 7-14).

3 Performance Evaluation

3.1 Simulation Environment

We simulate PATM by combining with the Dynamic Source Routing protocol (DSR), which is a reactive unicast ad hoc routing protocol, using NS-2 simulator [10]. The major control overhead of DSR is caused by Route Request packets (RREQs) which are flooded in the network in search of paths to the destinations. Therefore, we modify the Route Request phase such that every node rebroadcasts an RREQ packet if the node is not a host. As a result, hosts are excluded from intermediate nodes for a routing path.

We compare the performance of DSR with three modified DSR versions. Table 1 summarizes the different characteristics of the four routing protocols. We also compare the performance of PATM with another clustering algorithm, SPAN [12]. Both protocols run over DSR.

Table 1. Characteristics of Protocols

Protocol	With piggyback?	Topology adaptive?
DSR	No	No
DSR-PATM-1	No	No
DSR-PATM-2	Yes	No
DSR-PATM-3	Yes	Yes

We use the following metrics to show the performance of each protocol.

(1) *Normalized Control Overhead*: the total number of control packets divided by the total number of data packets delivered to destinations.
(2) *Delivery Ratio*: the total number of data packets delivered to destinations divided by the total number of data packets sent from sources.
(3) *Average Delay*: the average delay of all the data packets delivered to destinations.
(4) *Goodput*: the total number of data packets delivered to destinations during a simulation divided by the time span of the simulation.

3.2 Simulation Results

First, scenarios with different offered load are simulated where the number of CBR sessions varies from 15 to 60. The maximum speed of the nodes is 20m/s. In DSR-PATM-1 and DSR-PATM-2, the value of the interval T is set to 20s for all the nodes.

Fig. 7 shows the performance comparison under different metrics between the four protocols. It is apparent that the PATM-3 with piggybacking and adaptive update interval adjustment improves the delivery ratio, and reduces the routing overhead and average delay in most of the cases.

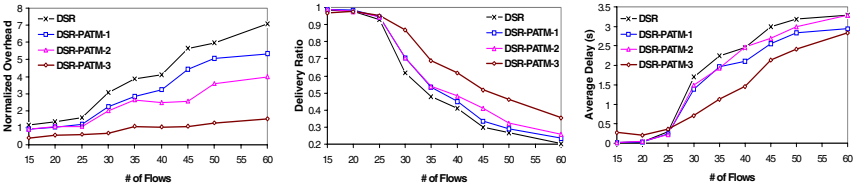


Fig. 7. Performance under Various Loads

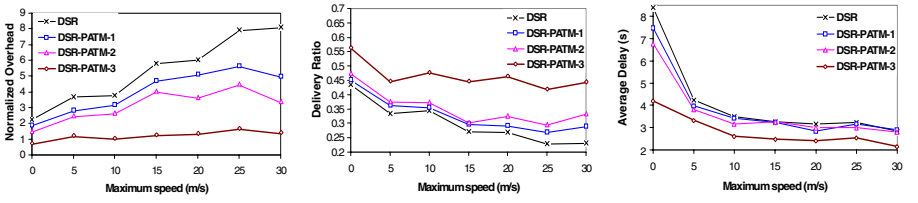


Fig. 8. Performance under Various Speeds

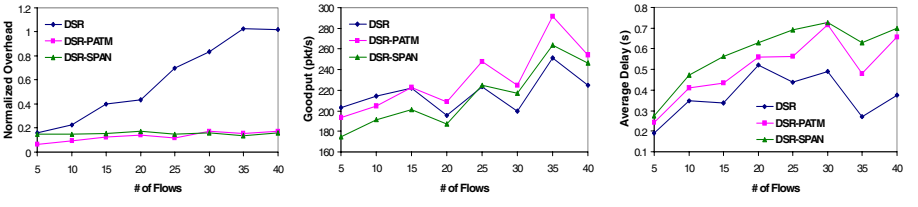


Fig. 9. Performance Comparison between PATM and SPAN

In the second set of simulations, we fix the number of CBR sessions to 50 and vary the maximum speed of nodes from 0 to 30m/s. Fig. 8 show the performance of the four protocols under different speeds. DSR-PATM-3 always performs the best among all the protocols, in both low mobility and high mobility scenarios.

Third, we compare PATM with SPAN. Here TCP flows are used to simulate data traffic. Each TCP flow lasts 900 seconds, which is the length of the whole simulation. Scenarios with different offered load are simulated where the number of TCP flows varies from 5 to 40. The maximum speed of the nodes is 20m/s. Fig. 10 shows the performance comparison, and PATM achieves smaller overhead and delay, and higher goodput.

4 Conclusions

We have presented PATM, a highly efficient topology management approach based on dynamic node priorities and network mobility. PATM builds a backbone of the original network for sufficient network connectivity and efficient data communica-

tion. We have applied it to routing protocols and have shown that it can reduce the control overhead significantly while improving the routing performance. Several optimizations have been applied in PATM such as update piggybacking, mobility-adaptive priority re-computation and topology information update. We show the application of PATM to the on-demand routing protocol DSR. Simulation studies demonstrate that PATM can reduce the routing overhead dramatically while improving the routing performance, in a variety of mobility scenarios with different traffic load.

References

1. Y. Xu, J. Heidemann, D. Estrin. Geography-informed energy conservation for ad hoc routing. Proc. of MobiCom 2001, Rome, Italy, pp. 70-84, July 2001.
2. T. Srinidhi, G. Sridhar, V. Sridhar, Topology management in ad hoc mobile wireless networks. Real-Time Systems Symposium, Work-in-Progress Session, Cancun, Mexico, December 3rd, 2003.
3. C.R. Lin, M. Gerla. Adaptive clustering for mobile wireless networks. IEEE Journal on Selected Areas in Communications, vol. 15, no. 7, pages 1265-1275, 1997.
4. C. Schurgers, V. Tsiatsis, S. Ganeriwal, M. Srivastava. Topology management for sensor networks: exploiting latency and density. Proc. of the 3rd ACM MobiHoc, Lausanne, Switzerland, June 9-11, 2002.
5. A. Cerpa, D. Estrin. ASCENT: Adaptive self-configuring sensor networks topologies. Proc. of IEEE INFOCOM, Jun. 2002.
6. L. Bao, J.J. Garcia-Luna_Aceves. Topology management in ad hoc networks. Proc. of the 4th ACM MobiHoc, Annapolis, Maryland, June 1-3, 2003.
7. Y. Xu, S. Bien. Topology control protocols to conserve energy in wireless ad hoc networks. Submitted for review to IEEE Transactions on Mobile Computing, January 2003. CENS Technical Report 0006.
8. M. Chatterjee, S.K. Das, D. Turgut. WCA: a weighted clustering algorithm for mobile ad hoc networks. ClusterComputing 5, pp. 193--204, 2002.
9. A. Amis, R. Prakash, T. Vuong, D.T. Huynh. MaxMin D-Cluster Formation in Wireless Ad Hoc Networks. Proc. of IEEE INFOCOM, March 2000.
10. NS notes and documentation. http://www.isi.edu/nsnam/ns.
11. Y. Yi, M. Gerla, T.J. Kwon. Efficient flooding in ad hoc networks using on-demand (passive) cluster formation. Proc. of the 3rd ACM MobiHoc, Lausanne, Switzerland, June 9-11, 2002.
12. B. Chen, K. Jamieson, H. Balakrishnan, Robert Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. Wireless Networks: 8 (5): 481-494, September 2002.

Practical and Provably-Secure Multicasting over High-Delay Networks*

Junghyun Nam[1], Hyunjue Kim[1], Seungjoo Kim[1], Dongho Won[1], and Hyungkyu Yang[2]

[1] School of Information and Communication Engineering, Sungkyunkwan University, Suwon-si, Gyeonggi-do 440-746, Korea
{jhnam, hjkim, dhwon}@dosan.skku.ac.kr, skim@ece.skku.ac.kr
[2] Department of Computer Engineering, Kangnam University, Yongin-si, Gyeonggi-do 449-702, Korea
hkyang@kangnam.ac.kr

Abstract. This paper considers the problem of authenticated key exchange in a dynamic group in which members join and leave the group in an arbitrary fashion. A group key exchange scheme for such a dynamic group is designed to minimize the cost of the rekeying operations associated with group updates. Although a number of schemes have attempted for many years to address this problem, all provably-secure schemes are inadequate in dealing with a dynamic group where group members are spread across a wide area network; their communication overhead for group rekeying is significant in terms of the number of communication rounds or the number of messages, both of which are recognized as the dominant factors that severely slow down group key exchange over a wide area network. In this paper, we propose an efficient key exchange scheme for this scenario and prove its security against an active adversary under the factoring assumption. The proposed scheme requires only a constant number of rounds while achieving low message complexity.

1 Introduction

A group key exchange scheme is designed to allow a group of parties communicating over an insecure public network like the Internet to establish a shared secret value called a session key. This group session key is typically used to facilitate standard security services, such as authentication, confidentiality, and data integrity, in various group-oriented applications like e.g. collaborative computing, audio/video conferencing, and distributed database. In other words, the essential goal of group key exchange protocols is to efficiently implement secure

* Seungjoo Kim is the corresponding author for this paper. This work was supported by the University IT Research Center Project funded by the Korean Ministry of Information and Communication.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 493–501, 2005.
© Springer-Verlag Berlin Heidelberg 2005

group communication channels over untrusted, open networks. The basic security requirement for a group key exchange scheme to achieve is the property referred to as (implicit) key authentication, whereby each member is assured that no one except the intended group members can obtain any information about the value of the session key. Therefore, the design of an efficient group key exchange scheme with key authentication is fundamental to network security and has recently received much attention as a consequence of the increased popularity of group-oriented applications [3, 16, 9, 14, 6, 15].

In this paper we focus on the problem of authenticated key exchange in a dynamic group, where current members may leave the group and new members may join the group at any time in an arbitrary manner. A group key exchange scheme for such a dynamic group must ensure that the session key is updated upon every membership change, so that subsequent communication sessions are protected from leaving members and previous communication sessions are protected from joining members. Although this can be achieved by running any authenticated group key exchange protocol from scratch whenever group membership changes, better handling of this dynamic membership problem has emerged as an important research goal toward efficient, scalable group rekeying [1, 7, 8, 12, 13, 16].

1.1 Related Work

In [9, 7, 8], Bresson et al. present the first formal security model for group key exchange, which is based on the work of Bellare et al. [4], and provide the first provably-secure protocols in this model. The initial work [9] assumes that group membership is static, whereas later works [7, 8] focus on the dynamic case. However, one obvious drawback of their scheme is that in case of initial group formation, its round complexity is linear in the number of users in the group. Moreover, the simultaneous joining of multiple users also takes a linear number of rounds with respect to the number of new users. Therefore, as group size grows large, this scheme becomes impractical particularly in wide area networks where the delays associated with communication are expected to dominate the cost of a group key exchange scheme.

Very recently, Katz and Yung [14] have presented a constant-round protocol which achieves both provable security and forward secrecy. This protocol, in its basic form, is based on the work of Burmester and Desmedt [10], and thus no efficiency gain over the Burmester-Desmedt protocol has been accompanied by its provable security. Indeed, this protocol adds one more round of n broadcasts for provable security, requiring in total three rounds of n broadcasts. Such a large number of message exchanges in one round is known as another negative factor that severely slows down group key exchange protocols in a wide area network setting. Furthermore, this protocol has to restart anew in the presence of any group membership change, because there is no known method to handle dynamic membership more efficiently for this protocol.

Most recently, in [6] Boyd and Nieto introduce another group key exchange protocol which is provably secure in the random oracle model [5] and requires

only a single round of communication to complete key exchange. But unfortunately, this protocol does not achieve forward secrecy even if its round complexity is optimal.

1.2 Our Contribution

The unsatisfactory situation described above has prompted this work aimed at designing an efficient and provably-secure key exchange scheme for a dynamic group where users communicate over a high-delay network environment. We provide a rigorous proof of security in the model of Bresson et al. [9, 7, 8] in which an adversary controls all communication flows in the network. The concrete security reduction we exhibit in the ideal hash model is tight; breaking the semantic security of our scheme almost always leads to solving the well-established factoring problem, provided that the signature scheme used is existentially unforgeable. Our group key exchange scheme also provides perfect forward secrecy. Namely, disclosure of long-term secret keys does not compromise the security of previously established session keys.

In wide area network environments, the main source of delay is not the computational time needed for cryptographic operations, but the communication time spent in the network. Moreover, the power of computers continues to increase at a rapid pace. We refer the reader to the literature [2, 13] for detailed discussions of comparison between the communication latency in wide area networks and the computation time for modular exponentiation. As the experiment results of [2] also indicate, it is widely accepted that the number of communication rounds and the number of exchanged messages are two most important factors for efficient key exchange over a wide area network.

Table 1. Complexity comparison among group key exchange schemes that achieve both provable security and forward secrecy

		Communication				Computation
		Rounds	Messages	Unicast	Broadcast	Exponentiations
[7]	IKE	n [1)]	n	$n-1$	1	$O(n^2)$
	Join	$j+1$	$j+1$	j [2)]	1	$O(jn)$
	Leave	1	1		1	$O(n)$
[14]		3	$3n$		$3n$	$O(n)+O(n^2 \log n)$ [3)]
Here	IKE	2	n	$n-1$	1	$O(n)$
	Join	2	$j+1$	j	1	$O(n)$
	Leave	1	1		1	$O(n)$

IKE: Initial Key Exchange

1) The number of users in a newly updated group
2) The number of joining users
3) $O(n^2 \log n)$: the number of modular multiplications

Table 1 compares the efficiency of our scheme given in Section 3 with other provably-secure schemes that provide forward secrecy [7, 14]. As for computa-

tional costs, the table lists the total amount of computation that needs to be done by users. As shown in the table, the scheme of [7] requires n communication rounds for initial key exchange which occurs at the time of group genesis, and j communication rounds for the rekeying operation that follows the joining of j new users. The protocol of [14], as already mentioned, requires n broadcast messages to be sent in each of three rounds, both for initial key exchange and for every group rekeying operation. In contrast, our scheme takes at most 2 communication rounds while maintaining low message complexity, in any of the three cases. Therefore, it is straightforward to see that our dynamic group key exchange scheme is well suited for networking environments with high communication latency. In particular, due to its computational asymmetry, our scheme is best suited for unbalanced networks consisting of mobile hosts with restricted computational resources and stationary hosts with relatively high computational capabilities.

2 Security Definitions

In this section, we first define what it means to securely distribute a session key within the security model given above and then explore the underlying assumptions on which the security of our scheme rests.

Authenticated Group Key Exchange. The security of an authenticated group key exchange scheme P is defined in the following context. The adversary $\mathcal{A}$, equipped with all the queries described in the security model, executes the protocols IKE1, LP1, and JP1 as many times as she wishes in an arbitrary order, of course, with IKE1 being the first one executed. During executions of the protocols, the adversary $\mathcal{A}$, at any time, asks a Test query to a fresh user, gets back an ℓ-bit string as the response to this query, and at some later point in time, outputs a bit b' as a guess for the secret bit b. Let $\mathsf{Good\text{-}Guess}$ be the event that the adversary $\mathcal{A}$ correctly guesses the bit b, i.e., the event that $b' = b$. Then we define the advantage of $\mathcal{A}$ in attacking P as

$$\mathsf{Adv}_P^{\mathcal{A}}(k) = 2 \cdot \Pr[\mathsf{Good\text{-}Guess}] - 1,$$

where k is the security parameter. We say that a group key exchange scheme P is secure if $\mathsf{Adv}_P^{\mathcal{A}}(k)$ is negligible for any probabilistic polynomial time adversary $\mathcal{A}$.

Secure Signature Schemes. We review here the standard definition of a digital signature scheme. A digital signature scheme $\Gamma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is defined by the following triple of algorithms:

- A *probabilistic key generation algorithm* $\mathcal{G}$, on input 1^k, outputs a pair of matching public and private keys (PK, SK).
- A *signing algorithm* $\mathcal{S}$ is a (possibly probabilistic) polynomial time algorithm that, given a message m and a key pair (PK, SK) as inputs, outputs a signature σ of m.

– A *verification algorithm* $\mathcal{V}$ is a (usually deterministic) polynomial time algorithm that on input (m, σ, PK), outputs 1 if σ is a valid signature of the message m with respect to PK, and 0 otherwise.

We denote by $\mathsf{Succ}_{\Gamma}^{\mathcal{A}}(k)$ the probability of an adversary $\mathcal{A}$ succeeding with an existential forgery under adaptive chosen message attack [11]. We say that a signature scheme Γ is secure if $\mathsf{Succ}_{\Gamma}^{\mathcal{A}}(k)$ is negligible for any probabilistic polynomial time adversary $\mathcal{A}$. We denote by $\mathsf{Succ}_{\Gamma}(t)$ the maximum value of $\mathsf{Succ}_{\Gamma}^{\mathcal{A}}(k)$ over all adversaries $\mathcal{A}$ running in time at most t.

Factoring Assumption. Let $\mathcal{FIG}$ be a factoring instance generator that on input 1^k, runs in time polynomial in k and outputs a $2k$-bit integer $N = p \cdot q$, where p and q are two random distinct k-bit primes such that $p \equiv q \equiv 3 \pmod 4$. Then, we define $\mathsf{Succ}_{N}^{\mathcal{A}}(k)$ as the advantage of adversary $\mathcal{A}$ in factoring $N = p \cdot q$ chosen from $\mathcal{FIG}(1^k)$. Namely,

$$\mathsf{Succ}_{N}^{\mathcal{A}}(k) = \Pr[\mathcal{A}(N) \in \{p, q\} \mid N(= pq) \longleftarrow \mathcal{FIG}(1^k)].$$

We say that $\mathcal{FIG}$ satisfies the factoring assumption if for all sufficiently large k, $\mathsf{Succ}_{N}^{\mathcal{A}}(k)$ is negligible for any probabilistic polynomial time adversary $\mathcal{A}$. Similarly as before, we denote by $\mathsf{Succ}_{N}(t)$ the maximum value of $\mathsf{Succ}_{N}^{\mathcal{A}}(k)$ over all adversaries $\mathcal{A}$ running in time at most t.

3 The Proposed Scheme

We now present a dynamic group key exchange scheme consisting of three protocols IKE1, LP1, and JP1 for initial group formation, user leave, and user join, respectively.

Let N be any possible output of $\mathcal{FIG}(1^k)$ and let $g \neq 1$ be a quadratic residue that is chosen uniformly at random in the set of quadratic residues in $\mathbb{Z}_N^*$, where $\mathbb{Z}_N^*$ is the multiplicative group modulo N. Then, we define the finite group $\mathbb{G}$, over which we must work, to be the cyclic subgroup of $\mathbb{Z}_N^*$ generated by g. For the rest of this paper, we denote by U_c the controller in a multicast group $\mathcal{MG}$, and by $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{\ell}$ a hash function modelled as a random oracle in the security proof of the scheme. For simplicity, we will often omit "mod N" from expressions if no confusion arises.

3.1 Initial Key Exchange: Protocol IKE1

Assume a multicast group $\mathcal{MG} = \{U_1, U_2, \ldots, U_n\}$ of n users who wish to establish a session key by participating in protocol IKE1. Then IKE1 runs in two rounds, one with $n-1$ unicasts and the other with a single broadcast, as follows:

1. Each U_i picks a random $r_i \in [1, N]$ and computes $z_i = g^{r_i} \bmod N$. $U_i \neq U_c$ then signs $U_i \| z_i$ to obtain signature σ_i and sends $m_i = U_i \| z_i \| \sigma_i$ to the controller U_c.

2. Upon receiving each message m_i, U_c verifies the correctness of m_i and computes $y_i = z_i^{r_c} \bmod N$. After receiving all the $n-1$ messages, U_c computes Y as $Y = \prod_{i \in [1,n] \setminus \{c\}} y_i \bmod N$ if n is even, and as $Y = \prod_{i \in [1,n]} y_i \bmod N$ if n is odd. U_c also computes the set $\mathcal{T} = \{T_i \mid i \in [1,n] \setminus \{c\}\}$ where $T_i = Y \cdot y_i^{-1} \bmod N$. Let $\mathcal{Z} = \{z_i \mid i \in [1,n]\}$. Then, U_c signs $\mathcal{MG} \| \mathcal{Z} \| \mathcal{T}$ to obtain signature σ_c and broadcasts $m_c = \mathcal{MG} \| \mathcal{Z} \| \mathcal{T} \| \sigma_c$ to the entire group.
3. Upon receiving the broadcast message m_c, each $U_i \neq U_c$ verifies the correctness of m_c and computes $Y = z_c^{r_i} \cdot T_i \bmod N$. All users in $\mathcal{MG}$ compute their session key as $K = \mathcal{H}(\mathcal{T} \| Y)$, and store their random exponent r_i and the set $\mathcal{Z}$ for future use.

To take a simplified example as an illustration, consider a multicast group $\mathcal{MG} = \{U_1, U_2, \ldots, U_5\}$ and let $U_c = U_5$. Then, in IKE1, the controller U_5 receives $\{g^{r_1}, g^{r_2}, g^{r_3}, g^{r_4}\}$ from the rest of the users, and broadcasts $\mathcal{Z} = \{g^{r_1}, g^{r_2}, g^{r_3}, g^{r_4}, g^{r_5}\}$ and $\mathcal{T} = \{g^{r_5(r_2+r_3+r_4+r_5)}, g^{r_5(r_1+r_3+r_4+r_5)}, g^{r_5(r_1+r_2+r_4+r_5)}, g^{r_5(r_1+r_2+r_3+r_5)}\}$. All users in $\mathcal{MG}$ compute the same key: $K = \mathcal{H}(\mathcal{T} \| Y)$, where $Y = g^{r_5(r_1+r_2+r_3+r_4+r_5)}$.

3.2 User Leave: Protocol LP1

Assume a scenario where a set of users $\mathcal{L}$ leaves a multicast group $\mathcal{MG}_p$. Then protocol LP1 is executed to provide each user of the new multicast group $\mathcal{MG}_n = \mathcal{MG}_p \setminus \mathcal{L}$ with a new session key. Any remaining user can act as the controller in the new multicast group $\mathcal{MG}_n$. LP1 requires only one communication round with a single broadcast and it proceeds as follows:

1. U_c picks a new random $r'_c \in [1, N]$ and computes $z'_c = g^{r'_c} \bmod N$. Using r'_c, z'_c and the saved set $\mathcal{Z}$, U_c then proceeds exactly as in IKE1, except that it broadcasts $m_c = \mathcal{MG}_n \| z_c \| z'_c \| \mathcal{T} \| \sigma_c$ where z_c is the random exponential from the previous controller.
2. Upon receiving the broadcast message m_c, each $U_i \neq U_c$ verifies that: (1) $\mathcal{V}(\mathcal{MG}_n \| z_c \| z'_c \| \mathcal{T}, \sigma_c, PK_c) = 1$ and (2) the received z_c is equal to the random exponential from the previous controller. All users in $\mathcal{MG}_n$ then compute their session key as $K = \mathcal{H}(\mathcal{T} \| Y)$ and update the set $\mathcal{Z}$.

We assume that in the previous example, a set of users $\mathcal{L} = \{U_2, U_4\}$ leaves the multicast group $\mathcal{MG}_p = \{U_1, U_2, \ldots, U_5\}$ and hence the remaining users form a new multicast group $\mathcal{MG}_n = \{U_1, U_3, U_5\}$. Also assume that U_5 remains as the controller in the new multicast group $\mathcal{MG}_n$. Then U_5 chooses a new random value r'_5, and broadcasts z_5, $z'_5 = g^{r'_5}$, and $\mathcal{T} = \{g^{r'_5(r_3+r'_5)}, g^{r'_5(r_1+r'_5)}\}$. All users in $\mathcal{MG}_n$ compute the same key: $K = \mathcal{H}(\mathcal{T} \| Y)$, where $Y = g^{r'_5(r_1+r_3+r'_5)}$.

3.3 User Join: Protocol JP1

Assume a scenario in which a set of j new users $\mathcal{J}$ joins a multicast group $\mathcal{MG}_p$ to form a new multicast group $\mathcal{MG}_n = \mathcal{MG}_p \cup \mathcal{J}$. Then the join protocol JP1 is run to provide the users of $\mathcal{MG}_n$ with a session key. Any user from the previous

multicast group $\mathcal{MG}_p$ can act as the controller in the new multicast group $\mathcal{MG}_n$. JP1 takes two communication rounds, one with j unicasts and the other with a single broadcast, and it proceeds as follows:

1. Each $U_i \in \mathcal{J}$ picks a random $r_i \in [1, N]$ and computes $z_i = g^{r_i} \bmod N$. $U_i \in \mathcal{J}$ then generates signature σ_i of $U_i \| z_i$, sends $m_i = U_i \| z_i \| \sigma_i$ to U_c, and stores its random r_i.
2. U_c proceeds in the usual way, choosing a new random r'_c, computing z'_c, Y, $\mathcal{T}$ and $K = \mathcal{H}(\mathcal{T} \| Y)$, updating the set $\mathcal{Z}$ with new z_i's, and then broadcasting $m_c = \mathcal{MG}_n \| z_c \| \mathcal{Z} \| \mathcal{T} \| \sigma_c$.
3. After verifying the correctness of m_c (including the verification by $U_i \in \mathcal{MG}_p \setminus \{U_c\}$ that the received z_c is equal to the random exponential from the previous controller), each $U_i \neq U_c$ proceeds as usual, computing $Y = z'^{r_i}_c \cdot T_i \bmod N$ and $K = \mathcal{H}(\mathcal{T} \| Y)$. All users in $\mathcal{MG}_n$ store or update the set $\mathcal{Z}$.

Consider the same example as used for LP1 and assume that a set of users $\mathcal{J} = \{U_2\}$ joins the multicast group $\mathcal{MG}_p = \{U_1, U_3, U_5\}$ to form a new multicast group $\mathcal{MG}_n = \{U_1, U_2, U_3, U_5\}$. Also assume that controller $U_c = U_5$ remains unchanged from $\mathcal{MG}_p$ to $\mathcal{MG}_n$. Then, U_5 receives $\{g^{r'_2}\}$ from the users in $\mathcal{J}$, and broadcasts z'_5, $\mathcal{Z} = \{g^{r_1}, g^{r'_2}, g^{r_3}, g^{r''_5}\}$ and $\mathcal{T} = \{g^{r''_5(r'_2+r_3)}, g^{r''_5(r_1+r_3)}, g^{r''_5(r_1+r'_2)}\}$ to the rest of the users, where r''_5 is the new random exponent of controller U_5. All users in $\mathcal{MG}_n$ compute the same key: $K = \mathcal{H}(\mathcal{T} \| Y)$, where $Y = g^{r''_5(r_1+r'_2+r_3)}$.

4 Security Result

Theorem 1. *Let the number of potential participants be bounded by a polynomial function $p_u(k)$ of the security parameter k. Let $\mathsf{Adv}_P(t, q_{se}, q_h)$ be the maximum advantage in attacking P, where the maximum is over all adversaries that run in time t, and make q_{se} Send queries and q_h random oracle queries. Then we have*

$$\mathsf{Adv}_P(t, q_{se}, q_h) \leq 2 \cdot \mathsf{Succ}_N(t') + 2p_u(k) \cdot \mathsf{Succ}_\Gamma(t''),$$

where $t' = t + O(q_{se} p_u(k) t_{exp} + q_h t_{exp})$, $t'' = t + O(q_{se} p_u(k) t_{exp})$, and t_{exp} is the time required to compute a modular exponentiation in $\mathbb{G}$.

In the following, we briefly outline the proof of Theorem 1[1]. The proof is divided into two cases: (1) the case that the adversary $\mathcal{A}$ breaks the scheme by forging a signature with respect to some user's public key, and (2) the case that $\mathcal{A}$ breaks the scheme without forging a signature. We argue by contradiction, assuming that there exists an adversary $\mathcal{A}$ who has a non-negligible advantage in

[1] The complete proof of the theorem is omitted here due to lack of space, and is given in the full version of this paper, which is available at *http://eprint.iacr.org/2004/115*.

attacking P. For the case (1), we reduce the security of scheme P to the security of the signature scheme Γ, by constructing an efficient forger $\mathcal{F}$ who given as input a public key PK and access to a signing oracle associated with this key, outputs a valid forgery with respect to PK. For the case (2), the reduction is from the factoring problem; given the adversary $\mathcal{A}$, we build an efficient factoring algorithm $\mathcal{B}$ which given as input $N = p \cdot q$ generated by $\mathcal{FIG}(1^k)$, outputs either p or q.

References

1. D.A. Agarwal, O. Chevassut, M.R. Thompson, and G. Tsudik: An Integrated Solution for Secure Group Communication in Wide-Area Networks. In Proc. of 6th IEEE Symposium on Computers and Communications, pp. 22–28, 2001.
2. Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik: On the Performance of Group Key Agreement Protocols. ACM Trans. on Information and System Security, vol.7, no.3, pp. 457–488, August 2004.
3. K. Becker, and U. Wille: Communication complexity of group key distribution. In Proc. of 5th ACM Conf. on Computer and Communications Security, pp. 1–6, 1998.
4. M. Bellare, D. Pointcheval, and P. Rogaway: Authenticated key exchange secure against dictionary attacks, Eurocrypt'00, LNCS 1807, pp. 139–155, 2000.
5. M. Bellare and P. Rogaway: Random oracles are practical: A paradigm for designing efficient protocols. In Proc. of 1st ACM Conf. on Computer and Communications Security (CCS'93), pp. 62–73, 1993.
6. C. Boyd and J.M.G. Nieto: Round-optimal contributory conference key agreement. PKC2003, LNCS 2567, pp. 161–174, 2003.
7. E. Bresson, O. Chevassut, and D. Pointcheval: Provably authenticated group Diffie-Hellman key exchange — the dynamic case. Asiacrypt'01, pp. 290–309, 2001.
8. E. Bresson, O. Chevassut, and D. Pointcheval: Dynamic group Diffie-Hellman key exchange under standard assumptions. Eurocrypt'02, pp. 321–336, 2002.
9. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater: Provably authenticated group Diffie-Hellman key exchange. In Proc. of 8th ACM Conf. on Computer and Communications Security, pp. 255–264, 2001.
10. M. Burmester and Y. Desmedt: A secure and efficient conference key distribution system. Eurocrypt'94, LNCS 950, pp. 275–286, 1994.
11. S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal of Computing, vol.17, no.2, pp. 281–308, 1988.
12. Y. Kim, A. Perrig, and G. Tsudik: Simple and fault-tolerant key agreement for dynamic collaborative groups. In Proc. of 7th ACM Conf. on Computer and Communications Security, pp. 235–244, 2000.
13. Y. Kim, A. Perrig, and G. Tsudik: Communication-efficient group key agreement. In Proc. of International Federation for Information Processing — 16th International Conference on Information Security (IFIP SEC'01), pp. 229–244, June 2001.

14. J. Katz and M. Yung: Scalable protocols for authenticated group key exchange. Crypto'03, LNCS 2729, pp. 110–125, August 2003.
15. J. Nam, S. Cho, S. Kim, and D. Won: Simple and efficient group key agreement based on factoring. In Proc. of the 2004 International Conference on Computational Science and Its Applications (ICCSA 2004), LNCS 3043, pp. 645–654, May 2004.
16. M. Steiner, G. Tsudik, and M. Waidner: Key agreement in dynamic peer groups. IEEE Trans. on Parallel and Distrib. Syst., vol.11, no.8, pp. 769–780, August 2000.

A Novel IDS Agent Distributing Protocol for MANETs[1]

Jin Xin, Zhang Yao-Xue, Zhou Yue-Zhi, and Wei Yaya

Key Laboratory of Pervasive Computing,
Department of Computer Science & Technology,
Tsinghua University, Beijing, China
jx01@mails.tsinghua.edu.cn

Abstract. Intrusion Detection Systems (IDSs) for Mobile Ad hoc NETworks (MANETs) is becoming an exciting and important technology in very recent years, because the intrusion prevention techniques can not satisfy the security requirements in mission critical systems. The proposed IDS architecture can be divided into two categories by the distributing form of IDS agents: fully distributed IDS and cluster-based IDS. The former has a high detection ratio, but it also consumes a cascade of energy. The latter has considered energy saving, but some hidden troubles of security exist in it. In this paper, we have proposed a novel IDS Agent Distributing (IAD) protocol for distributing IDS agents in MANETs. IAD protocol divides the whole network into several zones, selects a node subset from each zone, and runs IDS agent on the node in this subset. At the same time, IAD protocol can rectify the number of nodes running IDS agent according to the threat level of the network. Compared with the scheme that each node runs its own IDS, our proposed scheme is more energy efficient while maintaining the same level of detection rate. While compared with the cluster-based IDS scheme, our scheme is more flexible when facing the emergent situations. Simulation results show that our scheme can effectively balance the security strength and energy consuming in practice.

1 Introduction

With rapid development of MANET applications, security becomes one of the major problems that MANET faces today. MANET is much more vulnerable to attacks than wired networks, because the nature of mobility creates new vulnerabilities that do not exist in fixed wired networks. Intrusion prevention measures, such as encryption and authentication, can be used in MANET to reduce intrusions, but cannot eliminate them. In mission critical systems, which require strict secure communication, intrusion prevention techniques alone cannot satisfy the security requirements. Therefore, intrusion detection system (IDS), serving as the second line of defense, is indispensable for MANET with high security requirements.

In this paper, we present our progress in developing an IDS Agent Distributing (IAD) protocol for distributing IDS agents in MANET. In wired network, traffic monitoring is usually done at traffic concentration points, such as switches or routers. But in mobile ad hoc environment, there is no such traffic concentration point.

[1] Supported by the National 863 High-Tech plan (No. 2002AA111020).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 502 – 509, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Therefore, the IDS agents need to be distributed on the nodes in MANET. In addition, battery power is considered being unlimited in wired network, but MANET nodes typically have very limited batter power, so it is not efficient to make every node always run its IDS agent. The purpose of our scheme is to reduce the number of nodes running IDS agent, while maintain the same level of detection.

The rest of the paper is organized as follows. In section 2, we illustrate the motivation why we propose such an approach and some specific assumptions we rely on. Section 3 describes IAD protocol in detail. Simulation results are shown in Section 4. Section 5 concludes the paper.

2 Motivations and Assumptions

Extensive research has been done in this field and efficient IDS architectures have been designed for MANET. These architectures can be classified into two categories.

The first category is a fully distributed IDS architecture proposed in [1]. In this architecture, an intrusion detection module is attached to each node and each node in the network uses the local and reliable audit data source to participate in intrusion detection and response.

The second category is a cluster-based architecture. In [2], in order to address the run-time resource constraint problem, a cluster-based detection scheme is proposed. The whole network is organized as several clusters, each cluster elects a node as the clusterhead, and the clusterhead performs IDS functions for all nodes within the cluster.

There are some demerits on both of the two architectures. The first kind of architecture has a high detection ratio but it consumes a lot of power of each node. For the second kind of architecture, if a malicious node by any chance has been elected as the cluster head, it can launch certain attacks without being detected because it is the only node running IDS in the cluster. In addition, when mobility is high, the introduction of control overhead to create and maintain the cluster is unbearable. In one word, none of these two architectures considers both the effectiveness of IDS itself and the resource constrains of each mobile node at the same time.

3 IAD protocol

Aiming at solving this problem, we put the focus of our research on the combination of these two architectures and proposed the IAD (IDS Agent Distributing) protocol.

IAD protocol is based on the following assumptions:

1. Each node contains a unique and ordered identifier.
2. Neighbor information is always available.
3. Each node can overhear traffic within its transmission range.

IAD protocol consists of three sub-protocols: Neighbor Information Transmission (NIT) protocol, Cover Set Selection (CSS) protocol, and Monitoring Nodes

Adjustment (MNA) protocol. Node changes its state among the following states shown in Figure 1.

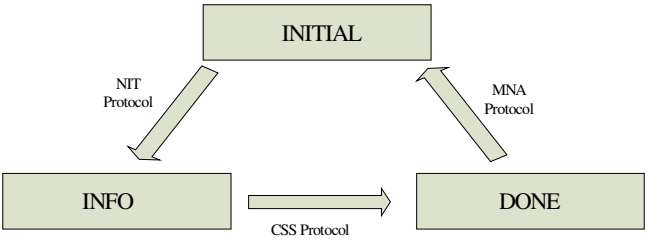


Fig. 1. State Changing in IAD Protocol

Initially, all nodes are in an INITIAL state. In that state, each node doesn't know the neighbor information of other nodes in the same zone. And, each node runs its own IDS agent. Once NIT protocol has finished, all nodes change their state from INTIAL to INFO, that is, each node has the information of other nodes' neighbor number. After CSS protocol has finished, a subset of nodes is selected, and the nodes in the subset continue running IDS agents while other nodes which are not included in the subset stop running IDS agent. All nodes enter into DONE state. If a node running IDS agent has detected an intruder, it will broadcast an Alarm message. Any node receiving this Alarm message will change its state back to INITIAL state again.

3.1 Neighbor Info Transmission Protocol

NIT protocol is responsible for transmitting a node's neighbor information to all other nodes in the same zone by a Neighbor Info Message (NIMsg), which data structure is defined in Table 1. For recording the NIMsg from other nodes, each node stores a Neighbor Info Table (NITbl) whose data structure is defined in Table 2.

Table 1. Data Structure of NIMsg

Field	Meanings
NodeAddr	Node address
ZoneID	Zone ID
SeqNum	Sequence number
NbrNum	Neighbor number of a node (in the same zone)
NbrChain	A chain that records the neighbors of a node

Table 2. Data Structure of NITbl

Field	Meanings
NodeAddr	Node address
NbrNum	Neighbor number of a node (in the same zone)
NbrChain	A chain that records the neighbors of a node

Without loss of generality, we discuss some zone *Z*. The topology map of zone *Z* is assumed as Figure 2. The number in the circle represents the neighbor number of the node. In the initial state, all nodes run their own IDS agent.

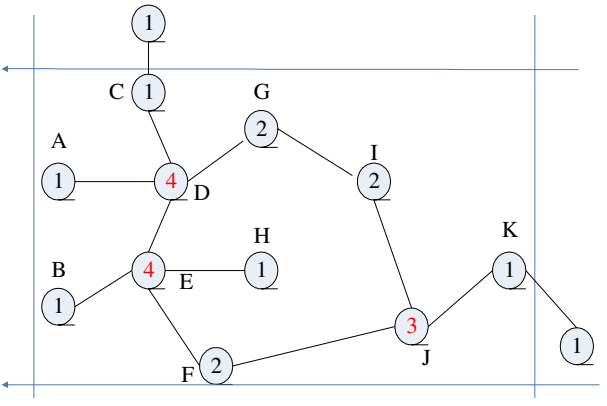


Fig. 2. Topology Map of Zone Z

Assume that *i*, *j* are two arbitrary nodes in zone *Z*. The concrete process of NIT protocol is described as following:

1. If none of the nodes has detected any intruder in a period T_1, every node will broadcast NIMsg.
2. When node *j* receives a NIMsg from node *i*, If $ZoneID_i = ZoneID_j$ and $SeqNum_i >= SeqNum_j$, then node *j* will do the following:
   (a) Restore the information of node *i* or update the old information of node *i* in its NITbl.
   (b) Forward this NIMsg by a probability *P* (*P*=0.7)[3].
3. Otherwise node *j* will discard this message.

Table 3. NITbl in node D

Node Address	Neighbor Number	Neighbor Chain
A	1	D
B	1	E
C	1	D
D	4	A->C->E->G
E	4	B->D->F->H
F	2	E->J
G	2	D->I
H	1	E
I	2	G->J
J	3	F->I->K
K	1	J

After NIT protocol has finished, all nodes in the zone have constructed a NITbl. For example, in Figure 2, node D will construct a NITbl shown as Table 3.

3.2 Minimal Cover Set Selection Protocol

We want to find out a subset of nodes, which satisfy that the nodes in the subset can overhear all the traffic in the zone. Then if only IDS agents run on these nodes, they can monitor all the traffic in the zone.

Assume that there is a set $A=\{a_1,a_2,\ldots\ldots,a_m\}$, a_i represents the node in network. And S_i represents all the neighbor nodes of node a_i, $S_i=\{a_i^1,a_i^2,\ldots\ldots,a_i^n\}$. If we can find out k nodes, which satisfy $\bigcup_{i=1}^{k} S_i = A$, then these nodes can overhear all the traffic in the zone. The key point is how to select this node subset. The above problem can be mapped to the classical minimal cover set problem in graph theory.

Minimal Cover Set Problem is a widely used problem among NP hard problems. There are many effective heuristic algorithms to solve it by now. Common heuristic algorithm is made up of one or more heuristic strategies. In general, the more the number of strategy, the more optimal the solution is. But more strategies also take more time to compute. Because of limited battery power in mobile nodes, and the high requirement for real time, we choose the greedy algorithm due to its simplicity and low complexity. Greedy algorithm always takes the best immediate, or local, solution while finding an answer. It finds the overall, or globally, optimal solution for some optimization problems, but may find less-than-optimal solutions for some instances of other problems. It never reconsiders this decision, whatever situation may arise later.

The concrete steps for finding a node subset by using greedy algorithm are in the following way:

1. Select a node from NITbl which has the most neighbors. If several nodes have the same neighbors, then the one which has the smallest address is selected.
2. Record the chosen node into the subset and delete the row which has the node in the NITbl.
3. Repeat step 1 and step 2, until the neighbors of nodes in the subset can cover all the nodes in the zone.

3.3 Monitoring Nodes Adjustment Protocol

When some node detects an intruder in the network, it will adjust the number of monitoring nodes using MNP protocol. The concrete process is as followings:

1. Once a node has detected an intruder, it will broadcast an Alarm message. The data structure of the Alarm message is shown in Table 4.
2. When any node receives this Alarm message, it will first determine the freshness of the Alarm message by SeqNum. If this Alarm message has already been received, the node will discard it. Otherwise, the node will perform following.
   - Record the AttackAddr and SeqNum field.
   - Start to run its IDS agent.
   - Forward this Alarm message.
   - Stop broadcasting NIMsg.

After all the nodes start their respective IDS agent, the network goes back to the fully distributed IDS architecture. We can use the method in [1] as the following intrusion respond method. If there is no intrusion in next period of T_1, the IAD protocol will begin its subset selection process again.

Table 4. Data Structure of Alarm message

Field	Meanings
AttackAddr	Address of the attacker
SeqNum	Sequence number

4 Simulation Results

We use a simulation model based on GloMoSim [4] to investigate the performance of the proposed approaches. In our simulation, the channel capacity of mobile hosts is set to the same value: 2 Mbps. We assume all nodes have the same transmission range of 250 meters. A free space propagation model with a threshold cutoff is used as the channel model. We use the Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs as the MAC layer protocol. It has the functionality to notify the network layer about link failures.

In the simulation, nodes move in a 1000*1000 meter region and we change the number of nodes from 30 to 100. The mobility model is the random waypoint model. The minimal speed is 5 m/s, and the maximal speed is 10 m/s. The pause time is 30 seconds. 5 source-destination pairs of CBR traffic and 3 source-destination of TCP traffic are used as the background traffic. All traffic is generated, and the statistical data are collected after a warm-up time of 300 seconds in order to give the nodes sufficient time to finish the initialization process.

Fig.3 compares the total number of nodes and the number of monitoring nodes under the same mobility level. From this figure, it can be seen that only half of the nodes are responsible to monitor neighbors in IAD protocol.

Fig.4 shows the average consumed power. In common conditions, the mode of wireless card can be divided into four kinds by the order of energy consumption: doze, idle, receiving and transmitting. Except doze, we call the other three modes as active state. In doze mode, Network Interface Card (NIC) neither sends nor receives signals, so this kind of mode is not fit for MANET. Feeney etc have tested the energy consumption of IEEE802.11 WaveLAN wireless network card produced by Lucent Corp. The result is shown in Table 5.

In GloMoSim simulator, wireless NIC is always under active state. The energy consumption consists of three parts: energy for sending data, energy for receiving data and energy in idle. When the amount of data to be sent and the sending power are fixed, the energy used to send data is fixed. In the fully distributed IDS architecture, when there is no data to be sent, the node is under receiving mode because every node needs to monitor the network, while in IAD protocol, when there is no data to be sent, the node will change its mode between idle and receiving according to its role. From the figure, we can conclude that IAD protocol can save about 10% of energy than that of fully distributed IDS architecture. When time goes on, IAD protocol will save more and more

energy. Especially, when traffic load is light, energy for sending data only accounts for a small ratio of energy consumption, then the power saving effect is more obvious.

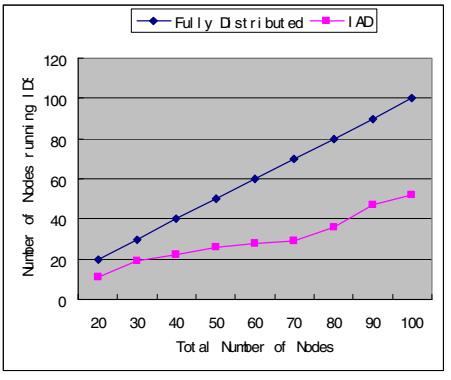


Fig. 3. Comparison of nodes

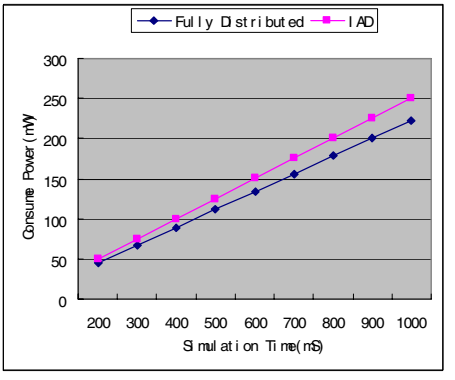


Fig. 4. Comparison of consumed power

Table 5. Data Structure of Alarm message

Mode	Actual Current	Actual Voltage	Referenced Current	Referenced Voltage
Doze	14mA	4.74V	9mA	5V
Idle	178mA		Null	
Receiving	204mA		280mA	
Transmitting	280mA		330mA	

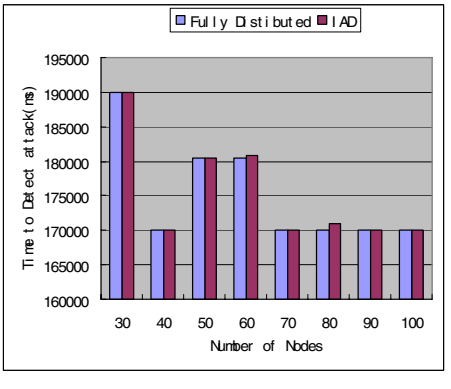


Fig. 5. Comparison of the time

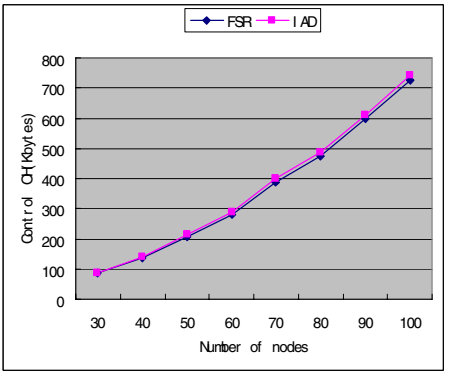


Fig. 6. Comparison of Control Overhead

Fig.5 compares the detection time between IAD protocol and the fully distributed scheme. We assume that there is one intruder sending a sequence of consecutive packets

constituting an attack to the destination. These packets are sent in a flow consisting of normal packets. Further, we assume that the nodes, which are a part of the intrusion detection subsystem, know this sequence of packets constituting the intrusion. The intrusion is considered being detected if this subsequence of attack packets pass through any of the nodes that constitute the intrusion detection subsystem.

In IAD protocol, since the nodes in the subset can overhear all the traffic of the network, the time spent by the IAD protocol to detect an intruder should be equal to the fully distributed scheme. Only when the attacked node moves out of the scope that the detecting nodes can overhear, the intruder may be detected later than fully distributed scheme. The figure shows that the IAD protocol can detect an attack almost as quickly as the fully distributed scheme. Even at the worse case, the IAD protocol only costs several more millisecond than the fully distributed scheme.

Figure 6 shows the additional control overhead introduced by IAD protocol. Because we implemented IAD protocol on FSR routing protocol, we compared the control overhead of original FSR routing protocol with that of IAD protocol. As shown in figure 6, IAD protocol only brings a small amount of additional control overhead. This part of control overhead is introduced by broadcasting NIMsg. Because 1) the period for broadcasting a NIMsg is much longer than routing update period, 2) the NIMsg is only broadcasted within the zone, 3) each node forwards this message only with a probability *P*, we can effectively control this part of overhead within a small scope.

5 Conclusion

Intrusion detection is an indispensable second wall of defense especially in any high-survivability network. Considering the limited computational and energy resources of mobile nodes, it is not efficient to make every mobile node always run IDS agent on itself. In this paper, we have proposed an IAD protocol for MANET. Its goal is to minimize the consumption of battery power and at the same time maintains an acceptable level of monitoring. It divides the whole network into several zones, selects a node subset that can overhear all the traffic from each zone, and all the nodes in the subset run IDS agents. In addition, it can rectify the detection level if intruders emerge. Simulation results show that the IAD protocol can implement the goals above efficiently.

Reference

[1] Y. Zhang and W. Lee, “Intrusion Detection in Wireless Ad Hoc Networks,” the 6th Annual Int’l Conf. on Mobile Computing and Networking (MobiCom’00), Boston, MA, Aug., 2000, pp. 275-283.

[2] Yian Huang and Wenke Lee,“A Cooperative Intrusion Detection System for Ad Hoc Networks,“In Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03), Fairfax VA, October 2003, pp. 135 – 147

[3] Z. Haas, J. Halpern, and L. Li, "Gossip-based ad hoc routing," in IEEE InfoCom Proceedings 2002, vol. 3, pp. 1707--1716, June 2002.

[4] X. Zeng, R. Bagrodia, and M. Gerla, “GloMoSim: a Library for Parallel Simulation of Large-Scale Wireless Networks,” Proc. of the 12th Workshop on Parallel and Distributed Simulations (PADS ’98), Banff, Canada,May 26-29, 1998, pp. 154-161.

ID-Based Secure Session Key Exchange Scheme to Reduce Registration Delay with AAA in Mobile IP Networks *

Kwang Cheol Jeong[1], Hyunseung Choo[1], and Sang Yong Ha[2]

[1] School of Information and Communication Engineering,
Sungkyunkwan University,
440-746, Suwon, Korea +82-31-290-7145
{drofcoms, choo}@ece.skku.ac.kr
[2] BcN Team IT Infrastructure Division,
National Computerization Agency, Korea

Abstract. Due to an increasing number of portable devices, a support for quality of service (QoS) and security becomes an main issue in Mobile IP networks. However Authentication, Authorization, and Accounting (AAA) protocol has inefficient authenticating procedures that limit its QoS. That is, a mobile node (MN) should be distributed new session keys whenever it performs a handoff. As computing power of mobile devices becomes greater, a key distribution using a symmetric key cannot guarantee the security. Hence, we employ an ID-based cryptography to intensify the security and when the MN moves to a new domain, a foreign agent (FA) reuses previous session keys encrypted by a public key for the fast handoff. Our proposed scheme reduces handoff delay and maintains high security by exchanging previous session keys between FAs. The performance results show that the proposed scheme reduces the latency up to about 63% compared to the previous ID-based AAA.

1 Introduction

Based on mobility as the essential characteristic for mobile networks, the Mobile IP de facto standard solution for use with the wireless Internet was developed by the Internet Engineering Task Force (IETF). Because the mobility implies higher security risks than static operations in fixed networks, there is a need to develop technologies which will jointly enable IP security and the mobility over wireless links, and thus adapting Mobile IPv6 to AAA protocol is suggested [2].

In the basic AAA protocol, AAA server distributes the session keys to MNs and agents to guarantee the security when they transmit data. Currently AAA protocol guarantees the security by using symmetric keys for information protection. Due to the drastically increasing computing power of devices, reliability

* This work was supported in parts by Brain Korea 21 and the Ministry of Information and Communication in Republic of Korea. Dr. H. Choo is the corresponding author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 510–518, 2005.
© Springer-Verlag Berlin Heidelberg 2005

on transmitting data based on symmetric keys can be threatened. Hence it is desirable to consider AAA protocol using asymmetric keys to enhance the security level. However when we consider Mobile IP networks which should support high mobility patterns, it seems hard to apply due to heavy operations. In the previous works whenever an MN arrives at a new domain, it performs a registration with its home network and after the MN is successfully authenticated and authorized, AAA server generates Mobile IP session keys (Mobile-Foreign, Foreign-Home, and Mobile-Home session key), but these processes need lots of operation time. In typical public key cryptography, the user's public key is explicitly encoded in a public key certificate. Therefore, the Public Key Infrastructure (PKI) model requires universal trust among the certificate issuers such as Certificate Authorities (CAs). This also has some well-known side effects such as cross-domain trust and certificate revocation. Moreover, PKI should maintain the structures such as CAs, Registration Authorities (RAs), and a directory servers containing certificates. Therefore, Shamir introduces an ID-based cryptography concept which simplifies certification management process [8].

In this paper, we propose an ID-based session key reuse mechanism which enhances the security in forwarding session keys and reduces the handoff time. In Section 2, an overview of the Mobile IP with AAA protocol, modern data encryption, and Identity (ID)-based cryptography are presented. We discuss the proposed ID-based session key reuse mechanism in Section 3. After that its performance is evaluated with previous methods in Section 4. Finally we conclude the paper in Section 5.

2 Related Works

2.1 AAA Protocol in Mobile IP

Within the Internet, an MN in an administrative area called a home domain often needs to use resources provided by another administrative zone called a foreign domain. An agent in the foreign domain that attends to the MN's request is likely to require that the MN provide some credentials that can be authenticated before the access to foreign resources. The agent may not have direct access to the data that is needed to complete the transaction. Instead, the agent is expected to consult a foreign AAA server (AAAF) in the same foreign domain in order to obtain the proof that the MN has acceptable credentials. Since the agent and the AAAF are part of the same administrative domain, they are expected to have security relationships that enable to transact information securely.

Since the AAAF itself may not have enough information to verify the credentials of the MN, it is expected to configure the verification process of MN credentials with home AAA server (AAAH). Once the authorization has been obtained by the AAAF and the authority has notified the agent for the successful negotiation, the agent can provide the requested resources to the MN [7]. AAA protocol operates based on the security associations which are defined by sharing the session keys [9].

2.2 Identity (ID)-Based Cryptography

The concept of ID-based encryptions and signatures is first introduced by Shamir in [8]. The motivation is to simplify certificate management and the essential idea of the ID-based cryptosystem is that any string ID consisting of $\{0,1\}^*$ can be the public key, and the author explains this by giving the example of the e-mail system [8]. The users should contact the Private Key Generator(PKG) to obtain a private key. Hence the ID-based cryptosystem does not need to access the public key directory and that means there is no need of PKI. Fig. 1 shows the comparison between a public key cryptosystem and an ID-based cryptosystem. For the secret communication in the public key cryptosystem, a sender should access to the public key directory for acquiring a public key. However in the ID-based cryptosystem, there is no need to access to the directory because an identity which is opened in the public channel is also a public key.

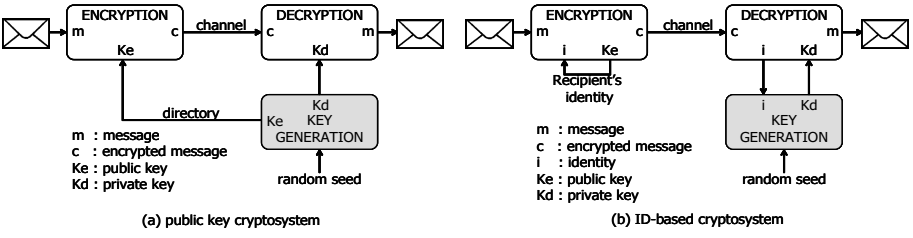


Fig. 1. Comparison between two cryptosystems

3 Session Key Reuse with ID-Based Cryptography

In this section, we describe the session key reuse mechanism with ID-based cryptography. In the proposed mechanism, we assume as follows:

- All nodes involved in Mobile IP with AAA protocol can calculate ID-based cryptography operations.
- Registration REPly (RREP) message includes the validity of MN without session keys.
- Private Key Generator (PKG) should have a master key to generate a private key corresponding to a public key for agents and MNs.

Fig. 3(a) and (b) show a Mobile IP registration procedure in AAA protocol [7] and a procedure of AAA protocol with an ID-based mechanism [4], respectively. The ID-based mechanism uses a digital signature for implementing mutual authentication which is one of main characteristics for the public key cryptosystem. In this case, because it is mutual authentication between the MN and Home Agent (HA), the authentication should occur at each entity (HA, FA, and AAAH) between the MN and the HA.

Fig. 3(c) shows the proposed AAA protocol with the ID-based mechanism which has modified procedure in the registration. The most remarkable difference

is that a new FA (nFA) receives previous session keys from an old FA (oFA). As you see in fig 3(b), the previous ID-based mechanism [4] should provide the signature verification at each entity between the MN and the HA because the MN should receive new session keys from the HA. As we all know, new session keys are issued by the AAAH and delivered through the HA. Meanwhile, the proposed mechanism provides the mutual authentication between oFA and nFA in the delivery of session keys. This minimizes the usage of the public key cryptography by using the previous session keys between oFA and HA at the registration reply from the HA and also provides the security from various attacks such as man in the middle attack. However, permanent using of issued previous session keys may cause another security problem. So there is a need to issue new session keys periodically based on timeout.

Registration Procedure in MIP with AAA Protocol

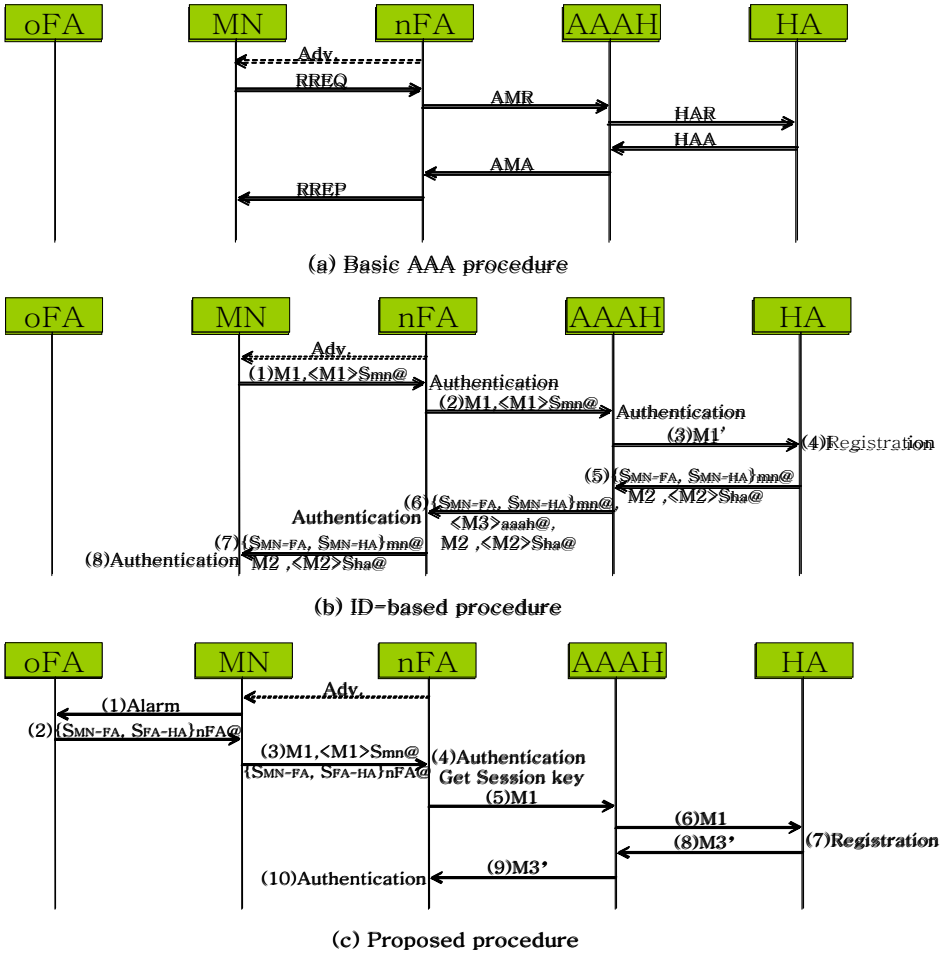
The following steps describe a process for the registration and authentication procedures in the ID-based mechanism. Refer to Fig. 3(b) and Table 1.

(1) When the MN detects the handoff is impended, it generates $M1$(corresponding to RREQ in Basic AAA procedure) and $<M1>S_{mn@}$ which is a signature for $M1$ based on the MN's ID, then sends them to an nFA. (2) The nFA authenticates the $M1$ based on the MN's ID and forwards messages to AAAH. (3) The AAAH also authenticates the $M1$ and sends $M1'$ ($M1$ with new session keys generated by AAAH) to HA. (4) After the HA registers a new CoA, (5) it encrypts two session keys S_{MN-FA} and S_{MN-HA} based on the MN's ID, and generates $M2$(corresponding to HAA in Basic AAA procedure) and its signature based on the HA's ID. Then it sends them to the AAAH. (6) The AAAH generates the signature of $M3$ and sends it along with all received messages to the nFA. (7) The nFA authenticates the $M3$ and sends all received messages except for $<M3>S_{mn@}$. (8) The MN authenticates the $M2$ based on the HA's ID and acquires two session keys. However, due to the absence of a security association between the MN and the nFA, it is vulnerable for some attacks at this point.

Table 1. Notation

Notation	Description
ID	Identity (e.g. e-mail address)
S_{ID}	Private Key for ID
aaah@	ID of AAAH
ha@	ID of HA
mn@	ID of MN
M	A message
$<M>S_{ID}$	Signature of M with S_{ID}
$\{M\}_{ID}$	Encryption of M with ID

And the following steps explain a process for the registration and authentication in our proposed ID-based mechanism. Refer also to Fig. 3(c) and Table 1.



RREQ: Registration Request message
RREP: Registration Reply message
AMR: AA-Mobile-Node Request message
AMA: AA-Mobile-Node Answer message
HAR: Home-Agent-MIP-Request message
HAA: Home-Agent-MIP-Answer message
AAAF: Authentication, Authorization and Accounting Server in Foreign Network
AAAH: Authentication, Authorization and Accounting Server in Home Network
M1: RREQ in ID-based Mechanism
M1': M1 with Session Keys
M2: HAA in ID-based Mechanism
M3: AMA with M2, $<M2>_{ha@}$
M3': AMA with Validity of the MN instead of Session Keys

Fig. 2. Registration procedures

(1) When the MN detects that the handoff is impended, it sends an alarm message which contains the nFA's ID to the oFA. (2) The oFA encrypts two session keys, S_{MN-FA} and S_{FA-HA} with a nFA's public key and then sends

it to the MN. (3) The MN creates $M1$(RREQ in ID-based) and its signature $<M1>S_{mn@}$, and sends them along with $\{S_{FA-MN}, S_{FA-HA}\}_{nFA@}$. (4) The nFA authenticates the $M1$ by verifying the $<M1>S_{mn@}$ based on the MN's ID and gets required session keys by decrypting based on its private key. (5) The nFA sends the $M1$ to AAAH. (6) The AAAH sends the $M1$ to HA. (7) The HA confirms a validity of the MN, registers the new CoA, and creates the encrypted $M3$ with a session key between the oFA and the HA. (8) The HA sends the $M3$ to AAAH. (9) The AAAH sends the $M3$ to nFA. (10) At this point, the nFA confirms the $M3$ that means the nFA has right session keys. Hence the nFA verifies the oFA which sends these session keys and also the oFA can verify the nFA which decrypts the encrypted message based on the nFA's ID.

4 Performance Evaluation

The values for the system parameters are directly from previous works, especially from [3] and [5]. And the time for Data Encryption Standard(DES), Message Digest 5(MD5), and Rivest-Shamir-Adlman(RSA) encryption and decryption is obtained from [10]. We compute the registration time with system parameters in Table 2. On the basic AAA procedure, the time for $RREQ_{MN-nFA}$ is computed based on the following simple estimation: 0.5 ms (MN processing time)+2 ms (propagation time in wireless links)+4.096 ms (message transmission time in wireless links)+0.088 ms (DES encryption and decryption)+0.0096 ms (MD5 operation)= 6.69 ms. The registration message size is assumed to 1024 bytes due to the RSA1024 operation [10]. Hence the message transmission time is obtained by multiplying the bit rate in wireless links and the message size.

- Basic AAA Method [7]
  $RREQ_{MN-nFA} + AMR_{nFA-AAAH} + HAR_{AAAH-HA} + HAA_{HA-AAAH} + AMA_{AAAH-nFA} + RREP_{nFA-MN} = 18.10\ ms$

- ID-based Method [4]
  $[M1, <M1>S_{MN@}, Auth.]_{MN-nFA}+$
  $[M1, <M1>S_{MN@}, Auth.]_{nFA-AAAH}+[M1, Registration]_{AAAH-HA}+$
  $[\{S_{MN-FA}, S_{MN-HA}\}_{mn@}, M2, <M2>S_{ha@}]_{HA-AAAH}+$
  $[\{S_{MN-FA}, S_{MN-HA}\}_{mn@}, <M3>_{aaah@}, M2, <M2>S_{ha@}, Auth.]_{AAAH-nFA}+$
  $[\{S_{MN-FA}, S_{MN-HA}\}_{mn@}, M2, <M2>S_{ha@}, Auth.]_{nFA-MN} = 37.62\ ms$

- Proposed Method
  $[\{S_{MN-FA}, S_{FA-HA}\}_{nFA@}]_{oFA-MN}+$
  $[M1, <M1>S_{MN@}, \{S_{MN-FA}, S_{FA-HA}\}_{nFA@}, Authen.]_{MN-nFA}+$
  $M1_{nFA-AAAH}+[M1, Registration]_{AAAH-HA}+$
  $[M3, Auth.]_{AAAH-nFA} = 23.12\ ms$

When we compare our proposed method to AAA with previous ID-based one [4], the registration time of the proposed one is reduced because the former uses the mutual authentication between the oFA and the nFA instead of the

Table 2. System parameters

Bit rates		**Processing time**	
Wire links	100 *Mbps*	Routers (HA,FA)	0.50 *ms*
Wireless links	2 *Mbps*	Nodes (MN)	0.50 *ms*
Propagation time		DES/MD5	0.044 *ms*/0.0048 *ms*
Wire links	500 μs	Signature creation	4.65 *ms*
Wireless links	2 *ms*	Signature verification	0.19 *ms*
Data size		RSA1024 encryption	0.18 *ms*
Message size	1024 *bytes*	RSA1024 decryption	4.63 *ms*

authentication between the MN and the HA. In the mutual authentication between the oFA and the nFA, session keys are delivered to the nFA from the oFA securely and therefore there is no need for the authentication at every related entity. Also the performance comparison shows that the proposed method takes a little bit more time than [7] because of using the public key cryptography, however it means the improved security level. The registration time required for the proposed method has drastically decreased compared to [4].

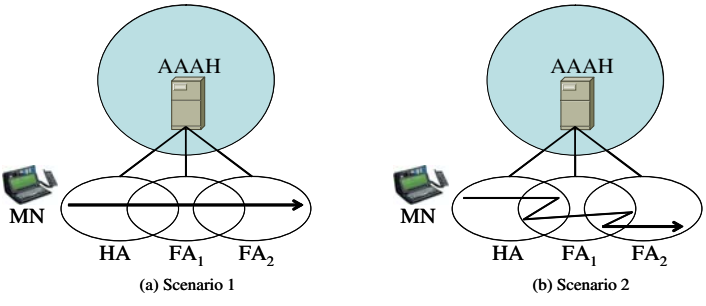


Fig. 3. Virtual network topology

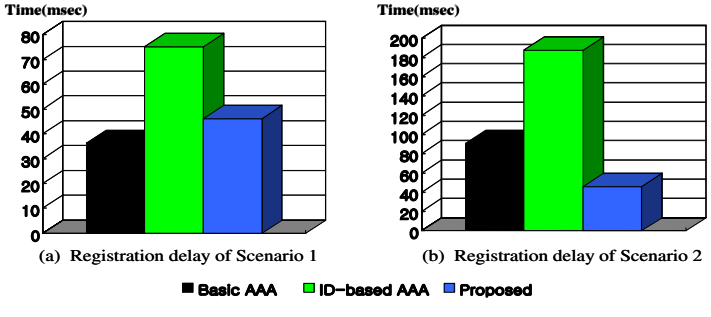


Fig. 4. Registration delays for three methods

As shown in Fig. 3, we have configured a simple virtual network topology for the comparison of various methods. In Fig. 3(a), suppose that an MN moves

directly from HA to FA_2. At this process, the MN performs handoff when it moves to a new area. Fig. 3(b) shows another scenario in the same virtual network topology. We assume that the MN moves zigzag within the overlapped area between adjacent cells in scenario 2. Fig. 4 shows the results. Fig. 4(a) represents a bar graph that shows the delay for the first scenario of the virtual network topology and Fig. 5(b) represents that of the second one. As you see in Fig. 4(a), our proposed scheme shows better performance than the ID-based scheme [4] even though it shows less performance than the basic AAA scheme [7]. And as you see in Fig. 4(b), our proposed scheme shows much better performance than previous two schemes. Even though the connection between the oFA and the MN is completely destroyed while performing the handoff, the proposed scheme shows the better performance since the oFA and the nFA share same session keys for the communication with the MN. Therefore MNs with high mobility patterns in overlapped areas, they do not need frequent authentication steps.

5 Conclusion

In this paper, we have proposed the session key reuse mechanism with ID-based cryptography. Based on the public key cryptography, this mechanism guarantees a higher level of security than the basic AAA mechanism [7] and has reduced registration time comparing to the AAA with the ID-based mechanism [4]. The result of the performance comparison also shows that the proposed mechanism is superior to AAA with the ID-based one [4] in terms of delay up to about 63% in the registration process. But due to heavy operations of public key cryptography, it takes a little bit more time than the basic AAA method. However, by minimizing the procedures which perform the public key cryptography, we can reduce the delay of the registration comparing to [7] while maintaining the similar level of security.

References

1. C. Boyd, "Modern data encryption," Electronic and Communication Engineering Journal, pp. 271–278, October 1993.
2. S. Glass, T. Hiller, S. Jacobs, and C. Perkins, "Mobile IP Authentication, Authorization, and Accounting Requirements," RFC2977, 2000.
3. A. Hess and G. Shafer, "Performance Evaluation of AAA/Mobile IP Authentication," Proceedings of 2nd Polish-German Teletraffic Symposium (PGTS'02), Gdansk, Poland, September 2002.
4. B.-G. Lee, D.-H Choi, H.-G. Kim, S.-W. Sohn, and K.-H. Park, "Mobile IP and WLAN with AAA authentication protocol using identity based cryptography," ICT2003 Proceedings, vol.3, pp. 597–603, February 2003.
5. J. McNair, I.F. Akyldiz, and M.D. Bender, "An inter-system handoff technique for the IMT–2000 system," INFOCOM 2000, vol. 1, pp. 203–216, March 2000.
6. C.E. Perkins, "IP Mobility Support," IETF RFC2002, October 1996.
7. C.E. Perkins, "Mobile IP and Security Issue: an Overview," Proceedings of 1st IEEE Workshop on Internet Technologies and Services, 1999.

8. A. Shamir, "Identity-based cryptosystems and signature schemes," Proceedings of Crypto '84, Springer-Verlag LNCS, vol. 196, pp. 46–53, 1985.
9. J. Vollbrecht, P. Cahoun, S. Farrell, and L. Gommans, "AAA Authorization Application Examples," RFC 2104, February 1997.
10. Wei Dai, "http://www.eskimo.com/weidai/benchmarks.html," Last modified: 13^{th} July 2003.

An Efficient Wireless Resource Allocation Based on a Data Compressor Predictor

Min Zhang[1], Xiaolong Yang [1], and Hong Jiang [2]

[1] Chongqing Univ. of Post and Telecommunication, Chongqing 400065, China
[2] Southwest University of Science and Technology, Sichuan Mianyang 621010, China
x_l_yang@sohu.com, zhangminyang@sina.com

Abstract. This paper discussed the resource allocation and reservation for wireless network, which is a challenging task due to the mobility uncertainty of user, Motivated from a rationale, i.e., a good data compressor should be a good predictor, we proposed a mobility prediction algorithm. Integrating the prediction algorithm into GC, a resource allocation scheme is also proposed. The numerical simulation results show that the time-complexity of our proposed scheme is worse, but it outperforms *Fixed-percent* and *ExpectedMax* in the QoS support effectiveness.

1 Introduction

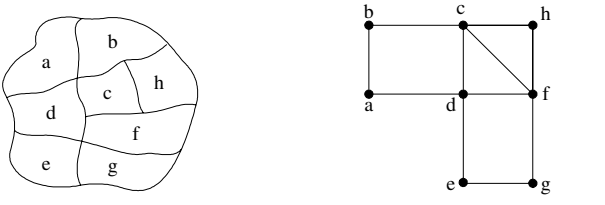
As known, the movement of the mobile users is greatly uncertain, which greatly impacts the efficiency of QoS schemes in wireless networks. In information theory, the Shannon entropy is a good method to quantitatively describe the uncertainty. In the same way, it can also scale the uncertainty of movement of mobile user. If the trajectory of movement of mobile user is regarded as a sequence of event, we can predict the next event by a certain data compression algorithm. Motivated by the theoretic bases and observations, this paper proposes a novel resource allocation and reservation scheme based on *Ziv-Lempel* algorithm, which is both theoretically optimal and good in practice.

2 The Description of the Model of User Mobility

Here, we use a generalized graph model to represent the actual wireless network (shown as Fig.1), where the cell shape and size may vary depending on many factors, such as the receiver sensitivity, the antenna radiation pattern of the base stations, and propagation environment, and the number of neighboring cells, which can be arbitrary but bounded and vary from cell to cell. An actual network can be represented by a bounded-degree, connected graph $G = (V, E)$, where the vertex-set V represents pairs of cells and the edge-set E represents the adjacency between pairs of cells. The example network shown in Fig. 1 can be modeled by the vertex set $V=\{a,b,c,d,e,f,g,h\}$ and the edge set $E=\{(a,d),(a,b),(b,c),\ldots,(e,g)$.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 519 – 526, 2005.
© Springer-Verlag Berlin Heidelberg 2005

In wireless networks, a call during its lifetime can be represented by a sequence of events *{N $H_1H_2SH_3H_nS...E$}*, where *N* denotes the event that a new call is admitted, H_n denotes the event of a mobile user *n*th handoff, *S* denotes the event of the call sojourning in the same cell and *E* denotes the call termination event. Note that in some cases, there are no handoff events during the lifetime of a call and thus no H_n in the sequence of events. According to the representation of the call events sequence, the trajectory of movement of mobile user can be also easily represented by a sequence of cells *{v}*, where *v( )* denotes the cells users handoff to. In the example network shown in Fig. 1, the trajectory of movement of a certain mobile user may be the cells sequence *{aabbbchfddedfch…}*. For a mobile user, its current location and the trend of movement can be described through these two sequences.



(a) An actual wireless cellular network which includes eight cell

(b) Its general graph representation

Fig. 1. Modeling an actual wireless cellular network

3 Mobility Predictions and Update Based on Data Compressor

For the mobility prediction processing, some literatures assume that the users independently handoff to its neighboring cells with equal probability, or assume that all of handoff events are independent and identically distributed. However, neither of them can exactly depict the mobility of users. Contrarily, the high-order Markov chain and the finite-context model are fairly reasonable. Of course, the order is higher, and the depictions of the actual movement can be more accurate, but the calculations of conditional probability and joint probability are also more difficult. Based on the definitions of entropy and conditional entropy in Ref. [6-7], the conditional entropy has a limit which equals to the per symbol entropy for a stationary stochastic process. So for each Markov chain and finite-context model, it is sure that there exists an appropriate order that depends on the mobility pattern. But its determination is a strenuous work, which root from the following factors. Firstly, its codeword is fixed-length because the conditional events of the *N*-order Markov chain are usually represented as $v_n|v_1v_2\ldots v_{n-1}$, which is equivalent to *N*-symbol codeword. Secondly, the relationship in the context is simplex and independent, and does not vary with the input sequences. Hence, for the mobility prediction processing, only the model with variable-length codeword and adaptive-context is reasonable and practicable. According to the analyses of Ref. [2-3], the symbol-based version of the *Ziv-Lempel* algorithm can become the best candidate for the model because it is both theoretically optimal and good in practice.

Essentially, the *Ziv-Lempel* algorithm is universal variable-to-fixed coding scheme, which parses the input string *S* into block-to-variable distinct but the shortest substrings *{s1, s2 s3,…}* in a greedy manner. For each $j \geq 1$, substring s_j without its last character is equal to some previous substring s_i, where $j > i \geq 0$.

Example 1: Let the symbols be *{a,b,c,d}*, and given an input string *S={aaababbbbbaabccddcbaaaa…}*, then the *Ziv-Lempel* encoder can parse it into the substrings *{a,aa,b,ab,bb,bba,abc,c,d,dc,ba,aaa,…}*.

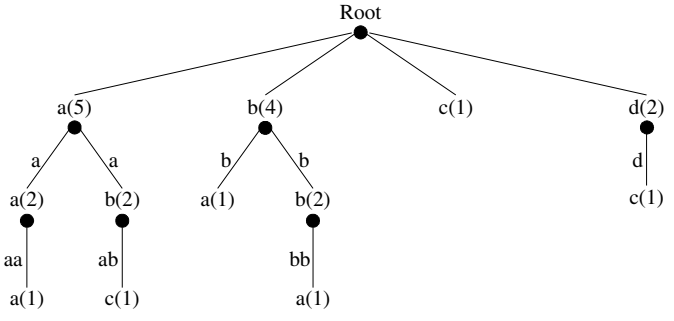


Fig. 2. The trie constructed in Example 1

The *Ziv-Lempel* coding process is interlaced with the learning process for the source characteristics. The key to the learning is a greedy decorrelating process, which implements by efficiently creating and looking up an explicit codeword dictionary. Because of the prefix property, substring parsed so far can be efficiently maintained in a trie [3], which can store statistics information for contexts explored besides representing the codeword dictionary. Fig. 2 shows the trie formed by Example 1. Through the trie, a new codeword can be easily created by concatenating a single symbol v to a parsed codeword s_i. As the parsing process progresses along with the string extending, larger and larger codeword accumulate in the dictionary. At the same time, the trie will be updated. Consequently, the estimates of conditional probabilities for larger contexts can be built up. Moreover, the learning capacity of the trie can be boosted up, and its prediction is also more precise.

The trajectory of movement of mobile user can be regarded as a substring in the symbol-based *Ziv-Lempel* algorithm. Similarly as shown in Fig. 3, we can also construct a mobility pattern predictor of the mobile user according to its mobility information base, which is equivalent to the trie. In natural, the predictor is a probability model based on *Ziv-Lempel* algorithm. When a new call is admitted, the predictor will set the current cell as a root of its mobility pattern, and update the probabilities of its possible events (including handoff and termination) during the call lifetime. When an event occurs in the sampling point, the predictor firstly judges whether it is the mobility pattern or not. If it is in the pattern, then the mobility pattern will be extended to the deeper layer, and be ready to the next prediction. Contrarily, a prediction fault is generated, and the mobility pattern and the context of current codeword are updated, as shown the red mark part in Fig. 3.

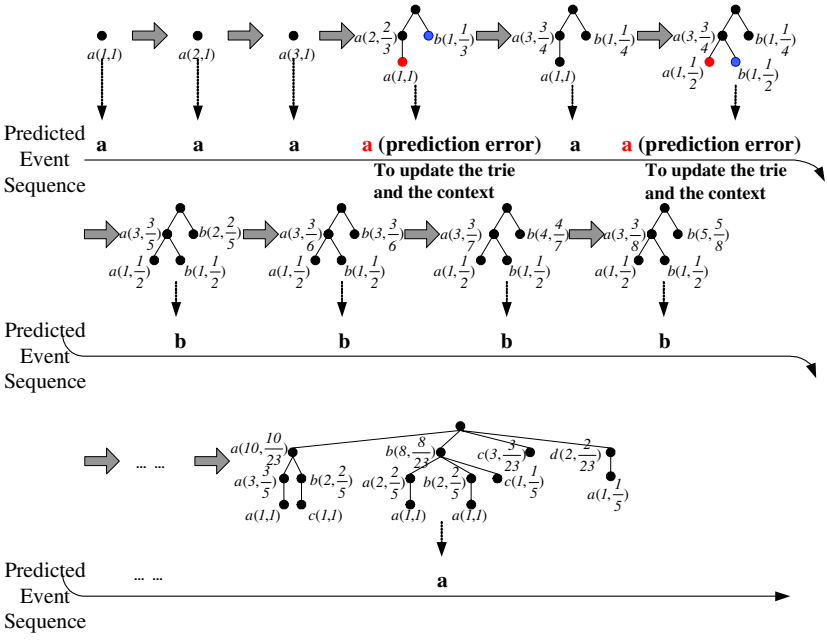


Fig. 3. The mobility prediction process of the symbol-based version of *Ziv-Lempel* encoder

The performance of the predictor can be evaluated by two metrics, i.e., the prediction fault rate and the expected fault rate. The former is defined as the ratio of the total number of prediction faults to the total number of events, and the latter is defined as the best possible fault rate achievable by any prediction algorithm which makes its prediction based only on the past mobility pattern history.

4 The Dynamic Resource Allocation and Reservation Scheme

Since forced call terminations due to handoff blocking are generally more objectionable than new call blocking, the handoff of a call should be treated in higher priority than the admission of new call, which is a default rule for service provider, and is also the premise of our scheme in this paper. To improve the QoS capacity of wireless networks, we must address the resource allocation and reservation scheme, which critical evaluating factors usually include handoff dropping probability, new call blocking probability, and the reserved resource utilization.

Among many schemes, the guard channels (*GC)* policy [1] and its revisions are simple, but they cannot satisfy the fluctuation of resource requirement due to user's mobility. However, their drawbacks can be overcome if their operations are based on the mobility predictor. So we can propose a scheme called prediction-based *GC*.

Before the prediction-based *GC* puts into effect, the most likely $cell_j$ (marked as *MLC*), which a mobile user may handoff to, must be firstly selected from the neighbor (marked as *Neighbor_Cell*) of current $Cell_i$ based on the mobility predictor in section

3. *Neighbor_Cel(Cell$_f$)* can be obtained from the non-zero items in the adjacency matrix (1). Note that the two cell set meet the following relation:

$$Cell_j \in MLC(Cell_i) \subseteq Neighbor_Cell(Cell_i) \tag{1}$$

Then we can pertinently allocate and reserve resource in *MLC(Cell$_i$)* for the call during its lifetime.

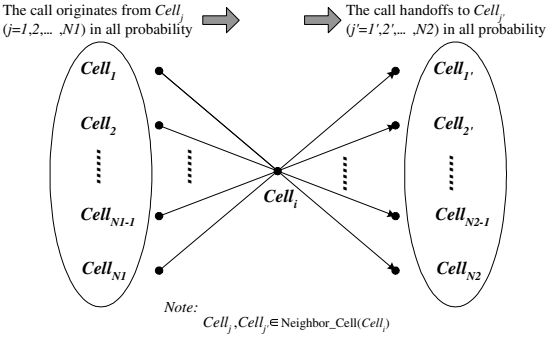


Fig. 4. The model of resource allocation and reservation during the call lifetime

As shown in Fig. 4, the call which handoffs to or originates from *Cell$_i$* is our discussion objective. Assumed that the call comes from N_1 source neighbor cells, and possibly handoffs to N_2 target neighbor cells. There exist n_i and n_i calls in each source and target neighbor cells, respectively. According to the above mobility predictor, we can get the context (i.e., transition probability) of the handoff event from *Cell$_j$* to *Cell$_i$* (from *Cell$_i$* to *Cell$_{j'}$*). The resource requirement in *Cell$_i$* will fluctuate along with the calls continuously arrival at and departure from *Cell$_i$*, which is represented as follows:

$$\Delta BW(N1,N2) = \sum_{l=1}^{N1} P_{l,i} \cdot \sum_{k=1}^{n_l} BW_k - \sum_{l'=1}^{N2} P_{i,l'} \cdot \sum_{k=1}^{n_{l'}} BW_k \tag{2}$$

where BW_k denotes the effective-bandwidth [**5**].

When a new call arrives at *Cell$_i$*, which resource requirement is *BW*, its operation strategy is represented by the following expression:

$$Available_BW_of_Cell_i > BW + \Delta BW \tag{3}$$

If the expression (4) holds, then the scheme admits the call, or else rejects it.

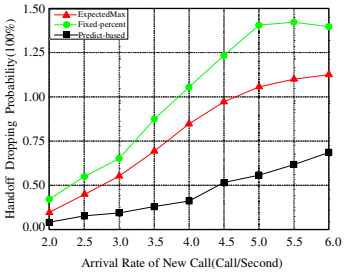
When *m* calls handoff to *Cell$_i$* while *n* calls terminate or handoff from *Cell$_i$* to other cells, its strategy is represented by the following expression:

$$Available_BW_of_Cell_i > BW_{Reserved} \tag{4}$$

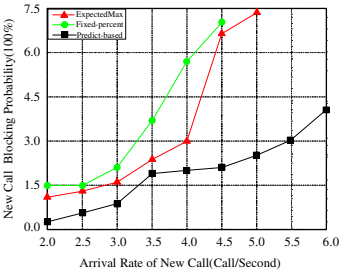
where $BW_{Reserved} = \Delta BW(N1+m, N2+n) - \Delta BW(N1, N2)$. If the expression (5) holds, then the scheme admits the calls handoffing to *Cell$_i$*, and reserves resources for them in advance, or else rejects the handoff requests.

5 Performance Analysis and Numerical Simulation

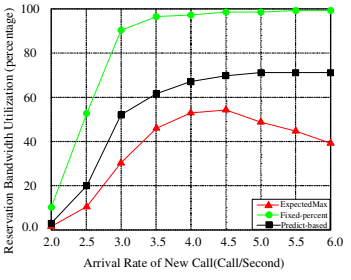
Our simulation network is a random Waxman-based network with 100 nodes (i.e., 100 cells). For simple, assumed the call creates in Poisson random process with average arrival rate λ and holding-time μ^{-1}, the total resource of each cell is 100 *unit*, and the resource requirement of each call uniformly distributes in the range [1, 20] *unit*. During the call lifetime, the trajectory of movement of mobile user is represented by the trace of a sub-graph of 100-node Waxman network.



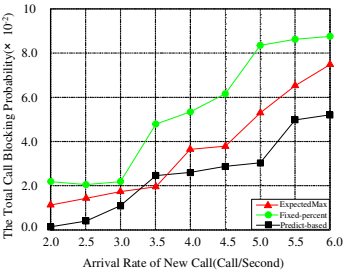
(a) The failure probability of call



(b) The blocking probability of new call



(c) The utilization rate of reserved bandwidth



(d) The total call blocking probability

Fig. 5. The QoS performance of our proposed scheme compared with other schemes

Here, we evaluate our proposed scheme by the comparison with *Fixed-percent* and *ExpectedMax* [**4**] from two aspects, i.e., the time-complexity, and the QoS performance which is depicted by the handoff dropping probability P_{hd}, the new call blocking probability P_{nb}, the reserved resource utilization, and the overall call blocking probability. For *Fixed-percent*, assumed that the reserved resource for handoff occupies *8%* of total resource in each cell.

Fig.5(a) and (b) illustrates the performances of three schemes in the handoff dropping and the new call blocking. As expected, P_{hd} and P_{nb} for the three schemes increase with the call arrival rate. But, both of P_{hd} and P_{nb} in our proposed scheme is the lowest, which benefits from the mobility prediction. Our proposed scheme considers the mobility of user during the resource reservation, and the reservation operations just aim at the predicted cells, i.e., $MLC(Cell_i)$. Obviously, the resource waste due to

the useless reservations in each cell is reduced. Therefore, P_{hd} and P_{nb} can be improved.

The reserved resource utilization is illustrated in Fig.5(c). When $\lambda<3.5$, it will rise sharply. But when $\lambda>3.5$, their distinctions will be more and more evident. In *Fixed-percent*, the reserved resource utilization will slowly rise and approach its limit *100%*. Contrarily in *Predict-based* and *ExpectedMax*, it will slightly decrease instead. It can be explained as follows. When λ increases, the constant resource reserved exclusively for the handoff process would be depleted, and cannot suffice the process of more handoff events. Hence, the utilization rate of its reserved resource certainly reaches *100%*. *Predict-based* and *ExpectedMax* reserve resource based on the mobility prediction. Moreover, these schemes exists some unavoidable prediction faults, which will appear frequently, and so incur much more invalid reservation operations when λ increases. Hence, it is impossible for *Predict-based* and *ExpectedMax* that the utilization rate is as high as *Fixed-percent*. As a whole, the utilization rate of *Predict-based* is better that of *ExpectedMax* When λ is high. The advantage comes from the better accuracy of mobility prediction based on Ziv-Lempel algorithm in *Predict-based*.

Obtained from the results in Ref. [**8**], the estimate of the overall call blocking probability in $Cell_i$ can be expressed as follows

$$P(\ Load_{Cell_i} \geq C\ ) \leq \left(\frac{Load_{Cell_i}}{C} \right)^C \cdot e^{C - Load_{Cell_i}} \tag{5}$$

where *Load* consists of the actual resource used by the existing calls in $Cell_i$, and the reserved resource in $Cell_i$ for some arriving calls. In the comparison of Fig. 6, we take the upper bound. As illustrated by Fig. 5, when $\lambda>3.5$, our proposed scheme distinctly outperforms *Fixed-percent* and *ExpectedMax*. Generally, the call arrival rate is more than *3.5-call/second* in actual wireless network. The result in Fig. 5 (d) shows that it is significant for our proposed scheme to improve the overall call blocking.

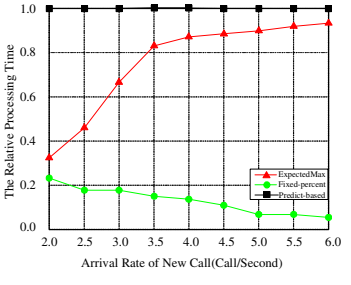


Fig. 6. The comparison of processing time, which of *Predict-based* acts as benchmark

As a whole, our proposed scheme will incur overhead as other schemes with mobility prediction. Here, we evaluate the overhead in terms of time-complexity. As shown by Fig. 6, the relative time-complexities of *Fixed-percent* and *ExpectedMax* are better than that of our proposed scheme. But when $\lambda>3.5$, the distinction between *Predict-based* and *ExpectedMax* will be more and more blurring.

6 Conclusions

Due to the mobility uncertainty of mobile user, it is a real challenge for wireless network to efficiently allocate and reserve resource. Firstly motivated from a rationale, i.e., a good data compressor should be a good predictor, this paper develops a mobility prediction algorithm based on the Ziv-Lempel algorithm, which is both theoretically optimal and good in practice. Theoretically, the prediction algorithm can predict not only to which cell a mobile user will handoff but also when the handoff will occur. Then, we propose an efficient re-source allocation and reservation scheme, called predict-based GC, which integrates the prediction algorithm into the guard channels (GC) policy. The simulation results show that the time-complexity of our proposed scheme (i.e., predict-based GC) is worse, but it outperforms Fixed-percent and ExpectedMax in the QoS support effectiveness.

References

1. E. C. Posner and R. Guerin, “Traffic policies in cellular radio that minimize blocking of handoff calls,” Proc. of 11th Teletraffic Cong., Kyoto, Japan, Sept. 1985
2. J. Ziv and A. Lempel, “Compression of individual sequences via variable-rate coding,” IEEE Transaction on Information Theory, Vol.24, No.5, pp.530-536, Sep. 1978.
3. J. S. Vitter and P. Krishnan, “Optimal prefetching via data compression”, Journal of the ACM, Vol.43, No.5, pp.771-793, September 1996.
4. P. Ramanathan, KM Sivalingam, P. Agrawal and S. Kishore, “Dynamic Resource Allocation Schemes During Handoff for Mobile Multimedia Wireless Networks”, IEEE Journal on Selected Areas in Communications, vol. 17, no. 7, pp. 1270-1283, July 1999
5. Anwar I. Elwalid, Debasis Mitra, "Effective bandwidth of general Markovian traffic sources and admission control of high speed networks", IEEE/ACM Transactions on Networking, Vol.1, No. 3,pp.329-343, June 1993
6. Bhattacharya, S.K.Das, “LeZi-Update: An Information Theoretic Approach to Track Mobile Users in PCS Networks”, Proc. of ACM/IEEE MobiCom'1999, pp.1-12, Aug. 1999.
7. Yixin Zhong, “The principle of information science” (the second edition), BUPT press, 1996
8. Andras Farago, “Blocking Probability Estimation for General Traffic Under Incomplete Information”, Proc. of ICCC’2000, pp.1547-1551, 2000.

A Seamless Handover Mechanism for IEEE 802.16e Broadband Wireless Access*

Kyung-ah Kim[1], Chong-Kwon Kim[2], and Tongsok Kim[1]

[1] Marketing & Technology Lab., KT, Seoul, Republic of Korea
{kka1, tongsok}@kt.co.kr
[2] School of Electrical Engineering and Computer Science,
Seoul National University, Seoul, Republic of Korea
ckim@popeye.snu.ac.kr

Abstract. Handover is one of the most important factors that may degrade the performance of TCP connections and real-time applications in wireless data networks. We developed a loss-free handover scheme called LPM (Last Packet Marking) for IEEE 802.16e-based broadband wireless access networks. By integrating MAC and network layer handovers efficiently, LPM minimizes the handover delay and eliminates packet losses during handover. Our performance study shows that LPM achieves loss-free packet delivery without packet duplication and increases TCP throughput significantly.

1 Introduction

At present, existing WLAN has several limitations such as narrow transmission coverage and the interference problem caused by using the ISM (Industrial, Scientific, Medical) band. In order to achieve a higher date rate and wider cell range, the IEEE committee initiated a project 802.16 which standardizes a WBA (Wireless Broadband Access) technologies. The 802.16 project [1] first specified the MAC and physical layers of broadband fixed wireless access system over the 10-66 GHz band. It provides up to several tens of Mbps by using fixed antennas and fixed MSS (Mobile Subscriber Station) in urban and suburban areas. In addition, the IEEE 802.16a [2] modifies the MAC physical layer specifications that facilitates the non-line-of-sight communications over the 2-11 GHz. Furthermore, the baseline specification is now being amended again for mobility enhancement (60km/h) under IEEE 802.16 TGe Mobile Wireless MAN [3], which also deals with transmission power control and power saving.

Compared to wired transmission systems, wireless systems suffer from limited bandwidth and error-prone transmissions. In addition, packet losses and service disruptions may occur during cell HOs (Handovers) in cellular networks.

* This work was supported in part by the Brain Korea 21 Project in 2004 and grant No. (R01-2004-000-10372-0) from the Basic Research Program of the Korea Science & Engineering Foundation.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 527–534, 2005.
© Springer-Verlag Berlin Heidelberg 2005

For the TCP, in particular, lost packets during HO severely degrade the TCP performance because of the sensitivity of the TCP mechanism for packet loss. The TCP retransmits the lost packets and slows down its transmission rate. Even worse, when multiple packets are lost during one congestion window, TCP senders enter into the slow-start phase and decrease the packet sending rate severely.

In this paper, we propose a seamless HO scheme called LPM (Last Packet Marking) for intra-domain mobility in IEEE 802.16e-based broadband wireless packet networks. LPM merges MAC and network layer HO procedures to reduce HO time and assures a safe HO by preparing the routing update before the real HO.

This paper is structured as follows: In section 2, we briefly preview the IEEE 802.16 TGe HO procedure and overview the micro-mobility protocols. In section 3, we describe a LPM method for seamless HO on IEEE 802.16e-based wireless broadband access networks. Then, we verify the LPM through computer simulations and present the conclusions in section 5.

2 Background

2.1 IEEE 802.16 TGe Handover

Let us explain the IEEE 802.16 TGe HO procedure briefly. A BS periodically broadcasts a neighbor advertisement management message to identify the network and define the characteristics of the neighbor BS to associated MSS (Mobile Service Station). An MSS may decode this message to find out information about the parameters of the neighbor BS. Each MSS will thus be able to scan the neighbor BS and measure the signal strength. If necessary, an MSS may select neighbor BS and prepare for the future HO by performing ranging and association procedures. Through ranging, the MSS can acquire the timing, power and frequency adjustment information of the neighbor BS. The target BS-MSS association information is reported to the serving BS.

The MAC layer (L2) HO is divided into two phases; the HO pre-registration phase and the real HO phase. During HO pre-registration, the target BS is selected and pre-registered with the MSS. However, the connection to the currently serving BS is maintained and packets may exchanged during the pre-registration phase. In the real HO, MSS releases the serving BS and re-associates with the target BS.

Now, let's look into the HO procedure in greater detail. Either an MSS or a serving BS may initiate the HO pre-registration. When the MSS initiates the HO pre-registration, it may indicate a possible target BS from a signal-quality point of view. After the MSS or the serving BS initiates HO pre-registration, the serving BS may acquire information from the neighbor BS regarding their capability of serving the requesting MSS. The serving BS may further notify the neighbor BS (through the backbone) of the impending HO. Fig. 1 shows an example of MAC layer HO call flow initiated by the MSS [3]. After receiving the

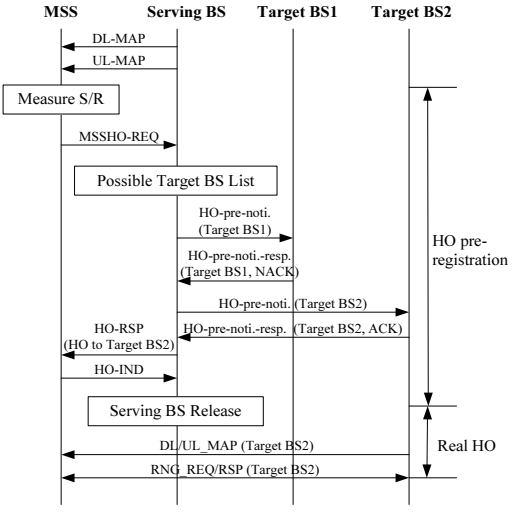


Fig. 1. Example of MAC layer HO call flow initiated by MSS

HO request (MSSHO-REQ) from the MSS, the serving BS sends an HO-pre-notification to the candidate target BSs, and the receiving party responds with an HO-pre-notification-response, which include ACK or NACK of impending HO for the MSS. Then, the serving BS selects the target BS and sends an HO-RSP message, including the target BS-ID, to the MSS. The MSS shall transmit an HO-IND message for final indication that it is about to perform a real HO. After the HO pre-registration phase, the real HO procedure is started. The serving BS releases the MSS and the MSS synchronizes with the target BS. Thereafter reauthorization and re-establishment of IP connectivity are taken.

2.2 Micro Mobility Protocols

At present, there are many efforts underway to provide Internet services on integrated wireless and wired networks. Supporting an efficient IP mobility is one of the major issues in constructing IP-based wireless access networks. Mobile users will expect the same level of service quality as wired network users. Even though the serving BS of the mobile user changes, IP connections should be continuously maintained transparently. The Mobile Internet Protocol [4] is the current standard for supporting global IP mobility in a simple and scalable manner. However, Mobile IP is targeted for static mobility support where a service continuation is not guaranteed.

A number of solutions like Cellular IP , HAWAII , Hierarchical Mobile IP that support dynamic mobility or cellular networking have been proposed. These approaches aimed to extend Mobile IP rather than to replace it. In order to handle the local movement of mobile hosts without interaction with the Mobile-IP-enabled Internet, they have adopted a domain-based approach. These intra-domain protocols are used for establishing and exchanging the state information

inside the wireless access networks, so as to get fast and efficient intra-domain mobility or micro-mobility control.

3 A Seamless Handover Mechanism - LPM

3.1 Wireless Access Network Model

The broadband wireless access network model that we propose uses a routing-based scheme for micro-mobility. The domain is formed by PARs (Packet Access Routers) and BSs (Fig. 2) in a tree structure and is connected to the Internet through the domain root PAR. The BSs use IEEE 802.16e for its wireless interface. For global mobility, the MSS is registered with the address of the gateway PAR on the HA (Home Agent). In the local domain, the MSS is identified by the IP address it uses in its home network. Each PAR maintains the routing cache in a soft-state manner through the periodic routing update information and upward data packets sent by MSSs.

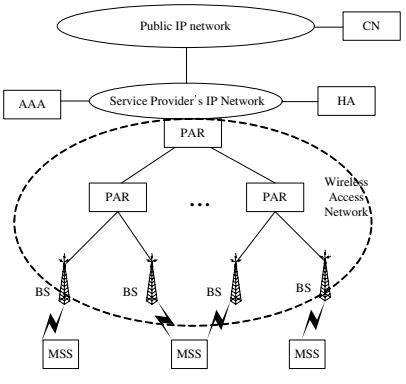


Fig. 2. Wireless Access Network Model

3.2 Proposed Seamless Hanover Protocol

In order to provide seamless mobility, the MAC layer (L2) HO and network layer (L3) HO should be integrated to minimize the impact on service performance. If the L3 HO is started after the L2 HO has been done, the IP connectivity should be broken for the time being until it is re-established. As a result, packet loss is inevitable. In our scheme, the L3 HO and L2 HO procedure progress concurrently so as to minimize HO time.

Each BS has BS-ID to BS-IP address mapping table of neighbor BSs in network initiation time. The proposed HO procedure is described in Fig. 3, in which the added procedures to the IEEE 802.16 TGe document are shown in bold and italic. Either the BS or the MSS can initiate HO. Then, the serving BS sends an HO-pre-notification (1) to the candidate target BS. The destination IP

address of the packet is on the BS-ID to BS-IP address mapping table in the serving BS. The MSS IP address should be added to the original message.

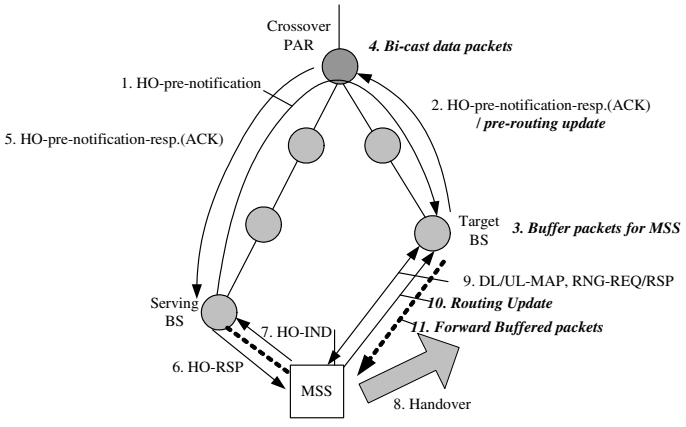


Fig. 3. LPM Handover procedure

When the target BS receives an HO-pre-notification message, it decides whether or not to accept this MSS for HO. Then, it sends an HO-pre-notification-response with ACK or NACK to the serving BS. When ACK is the response, the pre-routing update message is sent towards the gateway (2). The sender address of the pre-routing update is the IP address of the impending HO MSS. By pre-routing update message, a routing entry in the routing cache of PARs is added in the path through the target BS to cross-over PAR, which is the branching ancestor of the serving BS and the target BS. Then, the target BS prepares the buffer for the MSS (3), which assures the removal of packet loss during L2 HOs. When the crossover PAR receives the pre-routing update message, it bi-casts the data packets toward the MSS in the direction of both the serving and target BSs (4). The PAR that receives the pre-routing update can know whether it is a crossover or not by looking up the routing cache. If another different routing entry for the MSS is in the cache, then it is a crossover PAR. After the serving BS receives the HO-pre-notification-response (5), it exchanges HO-RSP (6) / HO-IND (7) with the MSS, including the target BS information. Then, the MSS starts the real HO. After the real HO, including ranging and association with the target BS (9), the MSS first sends the routing update message (10) towards the gateway to stop bi-casting of the crossover PAR. Then, the target BS forwards the buffered data for the MSS (11). After that, the MSS can continue its normal packet communication. In the proposed mechanism, the data packets received from the serving BS after the HO-pre-notification-response can also be received through the target BS. This is because the crossover PAR bi-casts the data packets just after receiving the HO-pre-notification-response and pre-routing update. Thus, the HO-pre-notification-response signals the time point, after which the data packets for the MSS are prepared on the target BS buffer. We termed our

proposal as LPM (Last Packet Marking), since the HO-pre-notification-response indicates that last packet before bi-casting has been received through the serving BS.

In cases where several target BSs send the HO-pre-notification-response with ACK, many crossover PARs bi-cast the data packets. At every bi-casting, just one mapping to the new leaf BS (just one downward link) is attached to the routing cache in the crossover PAR. The final routing tree is a subset tree of the full tree of wireless access networks. That is, in the worst case, where the serving BS sends the HO-pre-notification to all BSs in the access network, all BSs can receive data packets for the MSS after HOs. But after the routing update timeout, only the routing entry on the path that the MSS is attached to remains.

When the MSS postpones the real HO after receiving the HO-pre-notification-response, then the bi-casted packets should be received through the serving BS and also through the target BS, which results for the MSS to receive duplicated packets. Thus, the target BS should filter out the duplicated packets. However, the IP layer doesn't know the TCP sequence number. So, when the MSS sends the routing update just after the real HO, the information on the last packet received from the serving BS before the real HO can be sent to the target BS. The information is the resulting value of the hash function of (IP Header + Fixed Size IP Payload). When the target BS receives this hash value, it then finds the matched packet in the buffer and only forwards the following packets to the MSS to filter out the duplicated packets.

4 Simulation

4.1 Simulation Details

We used the micro-mobility extension for the ns-2 network simulator based on version 2.1b6. Since IEEE 802.16e is not yet implemented in ns2, we emulated it using an IEEE 802.11 wireless LAN. When no other MSS are contending for wireless resources, the MSS can stably communicate with the BS like in IEEE 802.16e. HO-pre-notification and response were exchanged between the serving and target BSs.

The simulation topology is shown in Fig. 4. The wireless access network is formed with PAR0-PAR5 and the BSs. The TCP source is CN (Correspondent Node) and the receiver is the MSS. All wired links in the access network are 10 Mb/s duplex links with a 5-ms delay. The CN and gateway (PAR0) link is set to 10 Mb/s duplex link with a 50-ms delay. MSS connects to BS using the ns-2 CSMA/CA 2Mb/s wireless link model. The link layer HO delay is set to 15 ms. An MSS starts the TCP connection with CN at time 3 and oscillates between BS1 and BS5 at a constant speed from time 5. The MH stays for about 10 seconds before moving to the next BS. The TCP Tahoe is used for TCP mechanism. The TCP window size is set to 20 and the packet size is 1024 Bytes.

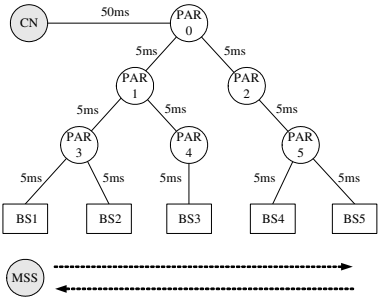
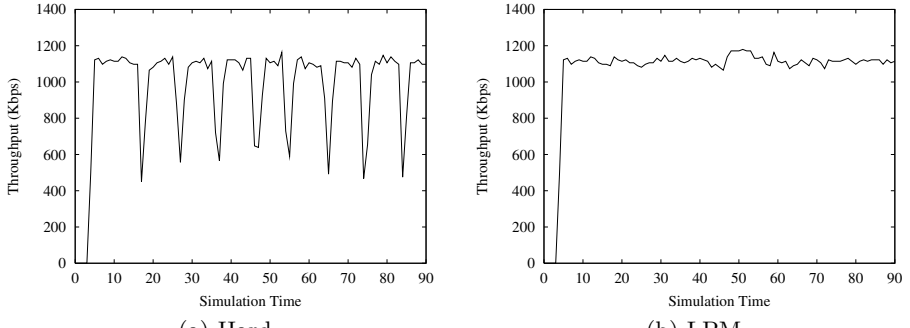


Fig. 4. Simulation Topology

4.2 Simulation Results

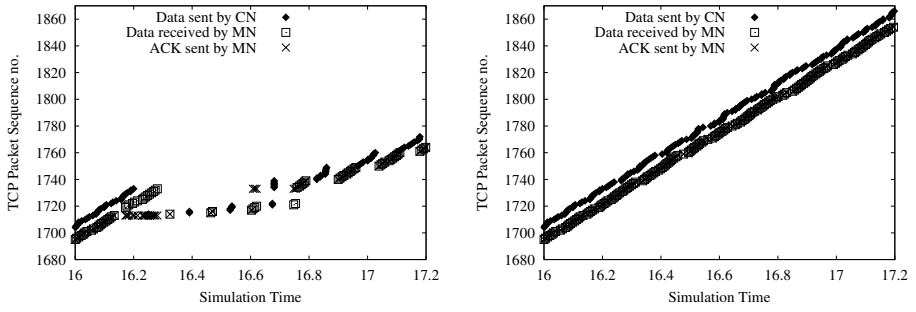
Figure 5 shows the TCP connection throughput as a function of time. The TCP throughput is measured every 1 second. We call the basic HO scheme as hard HO, in which L3 HO is started after the real HO. All hard HOs have abrupt glitches caused by lost packets. It is well known that a packet loss decreases the TCP performance significantly due to the TCP congestion control. On the other hand, LPM shows no throughput drops on any HO.



(a) Hard (b) LPM

Fig. 5. TCP throughput

The sender and receiver packet traces of the TCP connection from BS1 to BS2 HO is shown in Fig. 6. All other HO traces showed similar results. In hard HO, the real HO started at time 16.138 and finished at 16.155. Then, the L3 HO (routing update) is done from 16.155 to 16.172. The network layer HO time is proportional to the round-trip time from BS2 to crossover PAR (PAR3). TCP packet 1714 through 1718 was lost during this period. TCP restarts with slow-start from packet number 1714. But in LPM, no packet loss was observed. The HO-pre-notification message was sent at 16.134 and the response was received at 16.155. Real HO starts at 16.173 and ends at 16.188. During 16.155 to 16.173 the MSS receives bi-casted packets (1714 and 1718) from the serving BS. After



(a) Hard (b) LPM

Fig. 6. Sender and receiver traces of TCP connection

the real HO, the packet from 1714 to 1722 was buffered in the target BS. The target BS filtered out packets below 1719 to remove duplicate packets by hash value, included in the routing update from the MSS. Then, the BS forwarded the packets from 1719 to the MSS.

5 Conclusions

We have proposed a new handover scheme called LPM (Last Packet Marking) for micro-mobility in IEEE 802.16e-based broadband wireless packet networks. Through LPM, MAC and network layer handover procedures were done simultaneously to minimize the handover time. We studied the performance of LPM using computer simulation. Our simulation study showed that LPM is free from packet loss and duplication.

References

1. IEEE Standard 802.16, IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems (2001)
2. IEEE Standard 802.16a, Amendment 2: Medium Access Control Modifications and Additional Physical Layer Specifications for 2-11 GHz (2003)
3. IEEE 802.16 TGe Working Document, (Draft Standard) - Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands, 802.16e/D4, August (2004)
4. C. Perkins (ed.): IP Mobility Support for IPv4, Internet RFC 3344, Aug. (2002)

Fault Tolerant Coverage Model for Sensor Networks

Doina Bein[1], Wolfgang W. Bein[2], and Srilaxmi Malladi[3]

[1] School of Computer Science, University of Nevada Las Vegas, NV
siona@cs.unlv.edu
[2] School of Computer Science, University of Nevada Las Vegas, NV
bein@cs.unlv.edu
[3] Department of Computer Science, Georgia State University, GA
cscsrmx@cs.gsu.edu

Abstract. We study the coverage problem from the fault tolerance point of view for sensor networks. Fault tolerance is a critical issue for sensors deployed in places where are not easily replaceable, repairable and rechargeable. The failure of one node should not incapacitate the entire network.

We propose three 1 fault tolerant models, and we compare them among themselves, and with the minimal coverage model [8].

Keywords: Coverage, fault tolerance, smart sensors, sensor network.

1 Introduction

If all sensors deployed within a small area are active simultaneously, an excessive amount of energy is used, redundant data is generated, and packet collision can occur on transmitting data. At the same time, if areas are not covered, events can occur without being observed.

A *density control function* is required to ensure that a subset of nodes is active in such a way that *coverage* and *connectivity* are maintained. *Coverage* refers to the total area currently monitored by active sensors; this needs to include the area required to be covered by the sensor networks. *Connectivity* refers to the connectivity of the sensor network modeled as a graph: the currently active sensors has to form a fully connected graph such that the collected data can be relayed to the initiators (the nodes requesting data).

We study the coverage problem from the fault tolerance point of view. Fault tolerance is a critical issue depending on where the sensors are employed. Sensors coupled with integrated circuits, known as smart sensors, provide high sensing from their relationship with each other and with higher level processing layers. A smart sensor is specificaly designed for the targeted application [4]. Smart Sensors find their applications in a wide variety of fields such as military, civilian, bio-medical as well as control systems, etc. In military applications, sensors can track troop movements and help decide deployment of troops. In civilian

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 535–542, 2005.
© Springer-Verlag Berlin Heidelberg 2005

applications, sensors can typically be applied to detect pollution, burglary, fire hazards and the like.

It is clear that fault tolerance is improtant when maintaining survivability in such applications. The failure of one node should not incapacitate the entire network. Wireless body sensors implanted in the body must be energy efficient, utilize bandwidth, robust, lightweight and fault tolerant as they are not easily replaceable, repairable and rechargeable. Bio sensors need a dynamic, self-stabilizing network.

Motivation and Contributions. We are interested in the bio-medical domain where applications of sensors are relatively new. Sensors are already applied to monitor temperature level, glucose level, organs and its implants, and to detect external agents in the body in connection with cancer and other health abnormalities. We note that for such body chips to work properly, two-way communication between external computer controlling sensors and the actual sensors is needed. There may be tens and hundreds of sensors gathering data, thus wireless are preferable over wired sensors.

The goal of our paper is to propose several sensor array placement schemes, which are fault tolerant. Thus despite the presence of limted number of failed sensors, the system continues to function. We propose three 1 fault tolerant models, and we compare with each other and with the minimal coverage model[8].

Outline of the Paper. In Section 2, we review related work which has motivated our paper. Section 3 presents the various parameters for the sensor nodes and describes their relationship to our models, which are further described in Section 4. In Secton 5 we compare those models, and we finish with concluding remarks in Section 5.

2 Related Work

The nodes in a wireless environment are greatly dependent on the battery life and power. Therefore, minimizing energy consumption for the network while keeping its functionality is a major objective in designing a robust, reliable network. But sensors are prone to failures and disconnection. Only minimal coverage of a given region without redundancy would make such a network inattractive from a practical point of view. Therefore it is necessary to not only design for minimal coverage, on the other hand fault tolerance features must viewed in light of the additional sensors and energy used.

Given a sensor network deployed in a target area, [2] focused on finding whether each point of the area is covered by at least K sensors. [9] extends the problem further and focuses on selecting a minimum size set of sensor nodes which are connected in such a way that each point inside the area covered by the entire sensor network is covered by at least K sensors.

Starting from the uniform sensing range model [8], two models are proposed using sensors with different sensing ranges [7]. Variable sensing range is novel,

unfortunately both models are worse in terms of achieving a better coverage. Also, the second model in [7] requires (for some sensors) that the communication range to be almost six times larger than the sensing range, otherwise connectivity is not achieved.

A relay node, also called in the literature, *gateway* [1] or *application* node [5], acts as clusterhead in the corresponding cluster. In [3] a fault-tolerant relay node placement scheme is proposed for wireless sensor networks, and a polynomial time approximation algorithm is presented to select a set of active nodes, given the set of all the nodes.

In [6] the project of building a theoretical artificial retina made up of smart sensors is described. The sensors should form a tapered array that should rests on retina and produce electrical signals which are converted by the underlying tissue into chemical signals to be sent to the brain. The sensor array is therefore used for both reception and transmission in a feedback system. The challenges with these sensors are the wireless networking, distribution, placement and continuing operation of these sensors.

3 Preliminaries

Two parameters are important for a sensor node: the wireless communication range of a sensor r_C, and the sensing range r_S. They generally differ as values, and a common assumption is that $r_C \geq r_S$. Obviously, two nodes u and v, whose wireless communication ranges are r_{C_u}, respectively r_{C_v}, can communicate directly if $dist(u, v) \leq min(r_{C_u}, r_{C_v})$.

In [8], it is proven that if all the active sensor nodes have the same parameters (radio range r_C and sensing range r_S) and the radio range is at least twice of the sensing range $r_C \geq 2 \times r_S$, complete coverage of an area implies connectivity among the nodes. Therefore, under this assumption, the connectivity problem reduces to the coverage problem.

There is a trade-off between mimimal coverage and fault tolerance. For the same set of sensors, a fault tolerant model will have a smaller area to cover. Or, given an area to be covered, more sensors will be required, or the same number of sensors but with a higher values for the parameters.

A model is k fault tolerant if by removal of any k nodes, the network preserves its functionality. A k fault tolerant model for the coverage problem will be able to withstand k removals: by removing any k nodes, the covered region remains the same. A 0 tolerant model will not work in case of any removal of a node.

A straightforward approach is to either double the number of sensors in each point, or to double the sensor parameters for some sensors of the minimal coverage model to make it 1 tolerant. Similar actions can be taken for a 1 tolerant model to be 2 tolerant and so on. In order for a k fault-tolerant model to be worthwhile, it has to be better than the straightforward approach.

We propose three 1 fault tolerant models, and we compare them among each other, and with the minimal coverage model in [8].

4 Fault Tolerant Models

For all models, we assume the sensing range to be r, and we compare them among each other, and with the minimal coverage model [8]. In the first model, the basic structure is composed of four sensors arranged in a square-like structure of side r. In the second model, the basic structure is composed of six sensors arranged in a regular hexagon-like structure of side r. In the third model, the basic structure is composed of seven sensors arranged in a regular hexagon-like structure of side r, and the center of the hexagon as well. In these models the assumption that the communication range is greater than twice the sensing range guarantees the connectivity of the network.

4.1 Square Fault Tolerant Model

The basic structure for the first model is drawn in Figure 1(a).

(a) Four sensors in a square arrangement (b) Selected areas A, B, and C

Fig. 1. Square fault tolerant model

The square surface $S_4 = r^2$ is partitioned into an area covered by exactly two sensors S_{2s}^{square}, an area covered by exactly three sensors S_{3s}^{square}, and an area covered by exactly four sensors S_{4s}^{square}.

In order to calculate the values for those areas, let A, B, and C to be some disjoint areas as drawn in Figure 1(b). We observe that $S_{2s}^{square} = 4S_A$, $S_{3s}^{square} = 8S_B$, $S_{4s}^{square} = 4S_C$.

We can derive the following system of equations:

$$\begin{cases} S_A + 2S_B + S_C &= \frac{r^2}{4} \\ S_B + S_C + \frac{1}{4} &= \frac{\Pi r^2}{8} \\ 4S_B + 4S_C + S_A &= \frac{\Pi r^2}{3} - \frac{r^2\sqrt{3}}{4} \end{cases} \Rightarrow \begin{cases} S_A = r^2 - \frac{r^2\sqrt{3}}{4} - \frac{\Pi r^2}{6} \\ S_B = -\frac{r^2}{2} + \frac{r^2\sqrt{3}}{4} + \frac{\Pi r^2}{24} \\ S_C = \frac{r^2}{4} - \frac{r^2\sqrt{3}}{4} + \frac{\Pi r^2}{12} \end{cases}$$

$$\Rightarrow \begin{cases} S_{2s}^{square} = 4r^2 - r^2\sqrt{3} - \frac{2\Pi r^2}{3} \\ S_{3s}^{square} = -4r^2 + 2r^2\sqrt{3} + \frac{\Pi r^2}{3} \\ S_{4s}^{square} = r^2 - r^2\sqrt{3} + \frac{\Pi r^2}{3} \end{cases}$$

Therefore, given a 2D-region of dimension $(rN) \times (rM)$, with N and M strictly positive integers, we can derive the following results. The number of sensors required is $(N+1) \times (M+1)$. The ratio between the sensor area used and the area covered is $\frac{(N+1)(M+1)\Pi r^2}{NMr^2} = \frac{(N+1)(M+1)\Pi}{NM}$. The area covered by two sensors is $NMS_{2s}^{square} = NM(4r^2 - r^2\sqrt{3} - \frac{2\Pi r^2}{3})$.

The area covered by three sensors is $NMS_{3s}^{square} = NM(-4r^2+2r^2\sqrt{3}+\frac{\Pi r^2}{3})$.

The area covered by four sensors is $NMS_{4s}^{square} = NM(r^2 - r^2\sqrt{3} + \frac{\Pi r^2}{3})$.

4.2 Hexagon Fault Tolerant Model

The basic structure for the second model is drawn in Figure 2(a).

(a) Six sensors in a regular hexagon arrangement (b) Selected areas A and B

Fig. 2. Hexagon fault tolerant model

The hexagon surface $S_6 = \frac{3\sqrt{3}}{2}$ is partitioned into an area covered by exactly two sensors $S_{2s}^{hexagon}$, and an area covered by exactly three sensors $S_{3s}^{hexagon}$.

In order to calculate the values for those areas, let A and B be some disjoint areas as drawn in Figure 2(b). We observe that $S_{2s}^{hexagon} = 6S_A$ and $S_{3s}^{hexagon} = 6S_B$.

We can derive the following system of equations:

$$\begin{cases} S_A + S_B = \frac{r^2\sqrt{3}}{4} \\ \frac{1}{2}S_B + \frac{r^2\sqrt{3}}{4} = \frac{\Pi r^2}{6} \end{cases} \Rightarrow \begin{cases} S_A = \frac{3r^2\sqrt{3}}{4} - \frac{\Pi r^2}{3} \\ S_B = \frac{\Pi r^2}{3} - \frac{r^2\sqrt{3}}{2} \end{cases}$$

$$\Rightarrow \begin{cases} S_{2s}^{hexagon} = \frac{9r^2\sqrt{3}}{2} - 2\Pi r^2 \\ S_{3s}^{hexagon} = -3r^2\sqrt{3} + 2\Pi r^2 \end{cases}$$

Therefore, given a 2D region of dimension $(rN) \times (rM)$, with N and M strictly positive integers, we can derive the following results. The number of sensors required is $N\lceil\frac{2M}{sqrt3}\rceil$.

The ratio between the sensor area used and the area covered is $\frac{N\lceil\frac{2M}{sqrt3}\rceil \Pi r^2}{NMr^2} = \frac{\Pi\lceil\frac{2M}{sqrt3}\rceil}{M}$.

The area covered by two sensors is $\frac{1}{6}N\lceil\frac{2M}{sqrt3}\rceil S_{2s}^{hexagon} = \frac{1}{6}N\lceil\frac{2M}{sqrt3}\rceil(\frac{9r^2\sqrt{3}}{2} - 2\Pi r^2)$. The area covered by three sensors is $\frac{1}{6}N\lceil\frac{2M}{sqrt3}\rceil S_{3s}^{hexagon} =$

$$\frac{1}{6}N\lceil\frac{2M}{sqrt3}\rceil(-\overline{3}r^2\sqrt{3}+2\Pi r^2).$$

4.3 Improved 7-Node Model

We now consider instead of the minimal coverage model of three nodes, the seven-node model obtained by overlapping three three-node models (see Figure 3(a)), and we call it as the *improved model*.

(a) 7-node minimal coverage model (b) Fault tolerant improved 7-node model

Fig. 3. 7-node models

The minimal coverage model can be made 1 fault tolerant by modifying the sensing range of some nodes: one of the three sensors increases its sensing range from r to $r\sqrt{3}$. In this manner, when we overlap three such models we obtain the *improved 7-node model* (see Figure 3(b)).

Therefore, given a 2D region of dimension $(rN) \times (rM)$, with N and M strictly positive integers, we can derive the following results. The number of sensors required is $(\lfloor\frac{N}{\sqrt{3}}\rfloor + 1) \times (\lfloor\frac{2M}{3}\rfloor + 1)$.

5 Comparative Results

We consider the following models: the minimum coverage model [8], the improved model, the square model, and the hexagonal model. Let *f. t.* be a short form of *fault tolerant*.

We compare the models in terms of number of active nodes at a time required for covering a certain area, the covered area given a fixed number of nodes, and probability to function. Consider that all the sensors, independent of their sensing range, have the probability p to fail, $0 \leq p \leq 1$, therefore the probability

to function is $1 - p$. Also we assume that any two failures are independent one another.

In Table 1, we consider the following notations. *Covered area* denotes the area covered by the polygonal line formed by the sensors. *Fraction used* denotes the fraction of the sensor areas used for covering that area; this value aids in calculating the energy used for covering the region. *Efficiency* is defined as the ratio between the previous two values (the covered area and the fraction of the sensor area used), and denotes the efficiency of using a particular model. *Max. nodes to fail* denotes the maximum number of nodes that can fail and still the coverage is available over the number of nodes in the model. *Prob. to function* denotes the probability for the model to be functional.

The values for the probability functions in case of the square and hexagonal model from Table 1 follow. The probability to function in case of square model is $P_{square} = (1-p)^4 + 4p(1-p)^3 + 2p^2(1-p)^2 = (1-p)^2(1+2p-p^2)$. In case of the hexagonal model, the probability to function is $P_{hexa} = (1-p)^6 + 6p(1-p)^5 + 15p^2(1-p)^4 + 2p^3(1-p)^3 = (1-p)^2(1+3p+6p^2-8p^3)$.

Table 1. Comparisons among the four models

	Min. cov. 0 f. t.	Improved 1 f. t.	Square 1 f. t.	Hexagonal 1 f. t.
No. sensors	7	7	4	6
model covered area	$\frac{9\sqrt{3}r^2}{2}$	$\frac{9\sqrt{3}r^2}{2}$	r^2	$\frac{3r^2\sqrt{3}}{2}$
fraction used	$3\Pi r^2$	$5\Pi r^2$	Πr^2	$2\Pi r^2$
efficiency	$\frac{3\sqrt{3}}{2\Pi} \simeq 0.827$	$\frac{9\sqrt{3}}{10\Pi} \simeq 0.496$	$\frac{1}{\Pi} \simeq 0.318$	$\frac{3\sqrt{3}}{4\Pi} \simeq 0.413$
max. nodes to fail	0/6	6/7	2/4	3/6
prob. to function	$(1-p)^7$	$1-p+p(1-p)^6$	P_{square}	P_{hexa}

From Table 1 we observe that the minimal coverage model has the best efficiency, followed by the improved, hexagonal, and square model. Also, we observe that the hexagonal model has the highest probability to function, followed by the square, improved, and the minimal model.

6 Conclusion

We study the coverage problem from the fault tolerance point of view for sensor networks. Fault tolerance is a critical issue for sensors depending on where the

sensors are employed. The failure of one node should not incapacitate the entire network. Wireless body sensors have to be energy efficient, utilize bandwidth, robust, lightweight and fault tolerant as they are not easily replaceable, repairable and rechargeable.

We propose three 1 fault tolerant models, and we compare them among themselves, and with the minimal coverage model. We are currently working on algorithms to move sensors in order to preserve the network functionality when more than a fault occurs. If the network layout is composed by hundreds of such proposed models, in some cases sensors need to be moved to cover areas left uncovered by faulty or moving sensors.

References

1. G. Gupta and M. Younis. Fault-Tolerant clustering of wireless sensor networks. In *Proceedings of IEEE Wireless Communications and Networking Conf. (WCNC)*, pages 1579–1584.
2. C.F. Huang and Y.C. Tseng. The coverage problem in a wireless sensor network. In *ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 115–121, 2003.
3. B. Hao, J. Tang, and G. Xue. Fault-tolerant relay node placement in wireless sensor networks: formulation and approximation. In *IEEE Workshop on High Performance Switching and Routing (HPSR)*, pages 246–250, 2004.
4. A. Moini. Vision chips or seeing silicon. In *Department of Electrical and Electronics Engineering, University of Adelaide, Australia, http://www.iee.et.tu-dresden.de/iee/eb/analog/papers/mirror/visionchips/vision_chips/smart_sensors.html*, 1997.
5. J. Pan, Y.T. Hou, L. Cai, Y. Shi and S.X. Shen. Topology control for wireless sensor networks. In *Proceedings of ACM MOBICOM*, pages 286–299, 2003.
6. L. Schwiebert, S.K.S. Gupta, and J. Weinmann. Research challenges in wireless networks of biomedical sensors. In *ACM Sigmobile Conference*, pages 151–165, 2001.
7. J. Wu and S. Yang. Coverage issue in sensor networks with ajustable ranges. In *Intl. Conf. on Parallel Processing (ICPP)*, pages 61–68, 2004.
8. H. Zhang and J.C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. In *Proceedings of NSF Intl. Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, 2004.
9. Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. In *Intl. Conf. on Computer Communications and Networks (ICCCN)*, pages 373–378, 2004.

Detection Algorithms Based on Chip-Level Processing for DS/CDMA Code Acquisition in Fast Fading Channels *

Seokho Yoon[1], Jee-Hyong Lee[1], and Sun Yong Kim[2]

[1] School of Information and Communication Engineering, Sungkyunkwan University, 300 Chunchun-dong, Jangan-gu, Suwon, Kyunggi-do, 440-746, Korea
{syoon,jhlee@ece.skku.ac.kr}
[2] Department of Electronics Engineering, Konkuk University, 1 Hwayang-dong, Gwangjin-gu, Seoul 143-701, Korea
kimsy@konkuk.ac.kr

Abstract. In this paper, we propose various novel detection algorithms based on chip-level processing for direct sequence code-division multiple access (DS/CDMA) pseudo noise (PN) code acquisition in fast fading channels, wherein the fading process changes rapidly within the accumulation interval of the correlation samples between the locally generated and received PN codes. By applying the maximum-likelihood (ML) and locally optimal (LO) detection criteria to the correlation samples obtained on a chip-by-chip basis, both optimal and LO detection algorithms are derived. Both of these algorithms are found to include the conventional algorithm as a special case. Simpler suboptimal algorithms are also derived. Finally, numerical results show that the proposed algorithms can offer a substantial improvement over the conventional algorithm in fast fading channels.

1 Introduction

In direct sequence code-division multiple access (DS/CDMA) systems, rapid code acquisition is crucial, because data demodulation is possible only after code acquisition is performed. The basic unit in an acquisition system is a detector whose task is to identify, with a high degree reliability, the presence or not of alignment between the locally generated and received pseudo noise (PN) codes. The conventional detector which has been employed for code acquisition incorporates a detection algorithm based on the accumulation of the correlation samples between the locally generated and received PN codes. The accumulation of these samples is performed over an N (usually, $N \gg 1$) chips interval called the dwell time. Recently, with the allocation of higher frequency bands

* This work was supported by grant No. R01-2004-000-10690-0 from the Basic Research Program of the Korea Science & Engineering Foundation. Dr. S.Y. Kim is the corresponding author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 543–550, 2005.
© Springer-Verlag Berlin Heidelberg 2005

for mobile communications and/or the increasing relative velocity of user terminals with respect to either a base station or a satellite, some mobile channels (e.g., CDMA based mobile satellite channels) exhibit very fast fading such that the fading process may exhibit rapid changes within the dwell time [1]. In such a scenario, the fading fluctuations among the correlation samples become very significant and, consequently, the effective accumulation of the correlation samples cannot be achieved even when the locally generated and received PN codes are synchronized. Needless to say, this seriously deteriorates the performance of the acquisition systems. In [2] and [3], it was shown that the performance of acquisition systems employing the conventional detector in such a fast fading scenario can be improved by using a parallel search strategy and antenna arrays, as compared with that obtained using a serial search strategy and a single antenna, respectively. However, the performance is very sensitive to the fading rate and degrades substantially as the fading becomes faster, as a result of the conventional detector being employed. In this paper, novel detection algorithms which alleviate the fast fading effect are proposed for code acquisition. In the proposed algorithms, the correlation samples are efficiently processed at the chip level before their accumulation so that the fading fluctuations among the correlation samples can be alleviated and, consequently, the effective accumulation of the correlation samples over the dwell time can be achieved.

2 Observation Model

The typical structure of the PN code acquisition system is shown in Fig. 1. In the correlator, the received signal $r(t)$ is first down converted to baseband and then is correlated with the locally generated PN code. The complex baseband equivalent signal $r_l(t)$ of $r(t)$ may be expressed as

$$r_l(t) = \sqrt{P}\alpha(t)e^{j2\pi f_0 t}d(t-\tau T_c)c(t-\tau T_c) + w(t). \tag{1}$$

In (1), P is the transmitted signal power; $\alpha(t)$ is the (complex-valued) fading process; f_o is the frequency offset between the transmitter and receiver; $d(t)$ is the data waveform; T_c is the chip duration; $c(t)$ is the PN code waveform with a period of L chips; τ is the code phase normalized to T_c; and $w(t)$ is a zero-mean complex additive white Gaussian noise (AWGN) with one-sided power

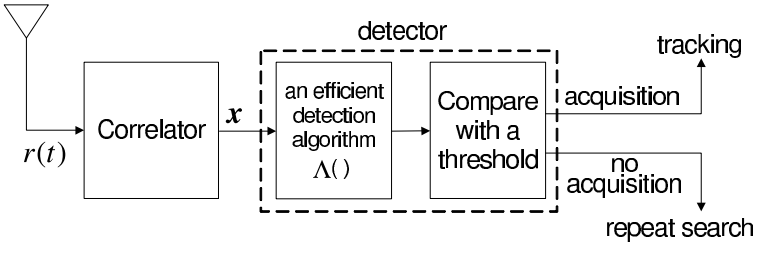


Fig. 1. Structure of PN code acquisition systems

spectral density N_0. The noise process $w(t)$ represents noise plus multiple access interference and is independent of $\alpha(t)$. In this paper, the fading process $\alpha(t)$ is assumed to have a Rayleigh fading envelope and a uniformly distributed phase, and to be wide-sense stationary. Then, $\alpha(t)$ can be modeled as a zero-mean complex Gaussian random process with the autocorrelation function given as $\phi(\Delta t) = E\{\alpha(t)\alpha^*(t+\Delta t)\}$, where $E\{\cdot\}$ and $*$ denote the statistical expectation and the complex conjugate, respectively, and $\phi(0) = 1$ due to normalization [4]. It is also assumed that there is a preamble for acquisition, so that no data modulation is present during acquisition, i.e., $d(t) = 1$. The baseband signal $r_l(t)$ is now correlated with the locally generated PN code and then sampled on a chip-by-chip basis during the dwell time of N chips. For simplicity, we assume that the system is chip synchronous and that the fading process $\alpha(t)$ is constant over one chip duration. Let hypotheses K and H correspond to in-phase and out-of-phase alignments, respectively, between the locally generated and received PN codes. Then, the kth correlation sample x_k, for $k = 1, 2, \cdots, N$, is given by

$$\begin{aligned} x_k &= \frac{1}{T_c}\int_{(k-1)T_c}^{kT_c} r_l(t)c(t-\hat{\tau}T_c)dt \\ &= \begin{cases} \sqrt{P}\alpha_k e^{j\pi\epsilon(2k-1)}\frac{\sin(\pi\epsilon)}{\pi\epsilon} + w_k, & \text{under } K \\ w_k, & \text{under } H, \end{cases} \end{aligned} \tag{2}$$

where $\hat{\tau}$ is the code phase (normalized to the chip duration T_c) of the locally generated PN code, ϵ is the frequency offset normalized to the chip rate T_c^{-1}, $\{\alpha_k\}_{k=1}^N$ are zero-mean complex Gaussian random variables with autocorrelation function $\phi(|m-n|T_c) = E\{\alpha_m\alpha_n^*\}$, and $\{w_k\}_{k=1}^N$ are zero-mean independent and identically distributed (i.i.d.) complex Gaussian random variables with variance $\sigma_w^2 = N_0 T_c^{-1}$. From (2), it is easy to see that $\{x_k\}_{k=1}^N$ are jointly complex Gaussian distributed and, thus, the pdf, $f_{\boldsymbol{x}}(\boldsymbol{x})$, of the correlation sample vector, $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)^{\mathcal{T}}$, with $(\cdot)^{\mathcal{T}}$ denoting the transpose, is given by

$$f_{\boldsymbol{x}}(\boldsymbol{x}) = \begin{cases} \frac{1}{\pi^N \det(\boldsymbol{\Gamma}_K)} \exp\left(-\boldsymbol{x}^{\mathcal{H}}\boldsymbol{\Gamma}_K^{-1}\boldsymbol{x}\right), & \text{under } K \\ \frac{1}{\pi^N \det(\boldsymbol{\Gamma}_H)} \exp\left(-\boldsymbol{x}^{\mathcal{H}}\boldsymbol{\Gamma}_H^{-1}\boldsymbol{x}\right), & \text{under } H, \end{cases} \tag{3}$$

where $\det(\cdot)$ and $(\cdot)^{\mathcal{H}}$ denote the determinant of a matrix and the Hermitian transpose, respectively. The elements located at row m and column n of the covariance matrices $\boldsymbol{\Gamma}_K = E\left\{\boldsymbol{x}\boldsymbol{x}^{\mathcal{H}}|K\right\}$ and $\boldsymbol{\Gamma}_H = E\left\{\boldsymbol{x}\boldsymbol{x}^{\mathcal{H}}|H\right\}$ are given by $P\phi(|m-n|T_c)e^{j2\pi\epsilon(m-n)}\frac{\sin^2(\pi\epsilon)}{\pi^2\epsilon^2} + \sigma_w^2\delta(m-n)$ and $\sigma_w^2\delta(m-n)$, respectively, where $\delta(\cdot)$ denotes the Kronecker delta function. Finally, a detection algorithm $\Lambda(\cdot)$ is performed with the correlation sample vector $\boldsymbol{x}$, and then its outcome is compared with a threshold. The conventional detection algorithm, denoted by $\Lambda_{\mathrm{C}}(\boldsymbol{x})$, is given by $\Lambda_{\mathrm{C}}(\boldsymbol{x}) = \boldsymbol{x}^{\mathcal{H}}\mathbf{1}_{N\times N}\boldsymbol{x} = \left|\sum_{k=1}^{N} x_k\right|^2$, where $\mathbf{1}_{N\times N}$ denotes an all-one matrix with size $N \times N$. It should be noted that in $\Lambda_{\mathrm{C}}(\boldsymbol{x})$ no processing is performed on the correlation samples before accumulation.

3 Optimal and Locally Optimal Detection Algorithms

3.1 Chip-Level Optimal (CLO) Detection Algorithm

A chip-level optimal (CLO) detection algorithm is derived using the maximum likelihood (ML) detection criterion. Using (3), the log likelihood ratio test can be written as follows: $\ln\left(\frac{f_{\boldsymbol{x}}(\boldsymbol{x}|K)}{f_{\boldsymbol{x}}(\boldsymbol{x}|H)}\right) = \ln\left(\frac{\det(\boldsymbol{\Gamma}_H)}{\det(\boldsymbol{\Gamma}_K)}\right) + \boldsymbol{x}^{\mathcal{H}}\left(\boldsymbol{\Gamma}_H^{-1} - \boldsymbol{\Gamma}_K^{-1}\right)\boldsymbol{x}$, in which the first term, $\ln\left(\frac{\det(\boldsymbol{\Gamma}_H)}{\det(\boldsymbol{\Gamma}_K)}\right)$, does not depend on $\boldsymbol{x}$. Thus, the CLO detection algorithm, denoted by $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$, may be obtained as

$$\Lambda_{\mathrm{CLO}}(\boldsymbol{x}) = \boldsymbol{x}^{\mathcal{H}}\left(\boldsymbol{\Gamma}_H^{-1} - \boldsymbol{\Gamma}_K^{-1}\right)\boldsymbol{x} = \boldsymbol{x}^{\mathcal{H}}\left(\sigma_w^{-2}\mathbf{I}_N - (\sigma_w^2\mathbf{I}_N + \gamma^2\mathbf{R}_{\boldsymbol{s}})^{-1}\right)\boldsymbol{x}, \quad (4)$$

where $\mathbf{I}_N$ denotes the identity matrix of size N, $\gamma = \sqrt{P}\frac{\sin(\pi\epsilon)}{\pi\epsilon}$, and $\mathbf{R}_{\boldsymbol{s}} = E\left\{\boldsymbol{s}\boldsymbol{s}^{\mathcal{H}}\right\}$ with $\boldsymbol{s} = (\alpha_1 e^{j\pi\epsilon}, \alpha_2 e^{j3\pi\epsilon}, \cdots, \alpha_N e^{j(2N-1)\pi\epsilon})^{\mathcal{T}}$. Under the assumption that the fading remains constant during the dwell time of N chips and that there is no frequency offset, $\mathbf{R}_{\boldsymbol{s}}$ can be simplified to $\mathbf{1}_{N\times N}$. Then, applying the Sherman-Morrison-Woodbury formula [5] to the term $(\sigma_w^2\mathbf{I}_N + \gamma^2\mathbf{R}_{\boldsymbol{s}})^{-1}$ in (4) gives $(\sigma_w^2\mathbf{I}_N + \gamma^2\mathbf{1}_{N\times N})^{-1} = \sigma_w^{-2}\mathbf{I}_N - \mathrm{C}\mathbf{1}_{N\times N}$, where $\mathrm{C} = \frac{\gamma^2}{\sigma_w^4 + N\gamma^2\sigma_w^2}$. Substituting this result into (4), we find that $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$ can be expressed as $\boldsymbol{x}^{\mathcal{H}}\mathbf{1}_{N\times N}\boldsymbol{x} = \left|\sum_{k=1}^{N} x_k\right|^2$, which is the conventional detection algorithm, $\Lambda_{\mathrm{C}}(\boldsymbol{x})$, mentioned in the previous section. This means that $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$ includes $\Lambda_{\mathrm{C}}(\boldsymbol{x})$ as a special case, wherein the fading remains constant during the dwell time and there is no frequency offset.

3.2 Chip-Level Locally Optimal (CLLO) Detection Algorithm

As shown in (4), $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$ requires the inversion of the $N \times N$ matrix. To derive an algorithm which is simple compared with $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$, we use the locally optimal (LO) detection criterion, which leads to the LO detection algorithm in signal detection theory. The LO detection algorithm is usually much easier to implement than the optimal detection algorithms, and yields the maximum outcome value when the signal-to-noise ratio (SNR) approaches zero [6]. From the generalized Neyman-Pearson's fundamental lemma [6] and (3), we can obtain the chip-level locally optimal (CLLO) detection algorithm as

$$\Lambda_{\mathrm{CLLO}}(\boldsymbol{x}) = \frac{1}{f_{\boldsymbol{x}}(\boldsymbol{x}|H)}\frac{d^{\nu} f_{\boldsymbol{x}}(\boldsymbol{x}|K)}{d\mu^{\nu}}\bigg|_{\mu=0} = \boldsymbol{x}^{\mathcal{H}}\mathbf{R}_{\boldsymbol{s}}\boldsymbol{x}, \quad (5)$$

where μ is a signal strength parameter (in this paper, we set $\mu = \gamma = \sqrt{P}\frac{\sin(\pi\epsilon)}{\pi\epsilon}$) and ν is the order of the first nonzero derivative of $f_{\boldsymbol{x}}(\boldsymbol{x}|K)$ at $\mu = 0$. It is noteworthy that, in contrast to $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$, $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ does not require matrix inversion. When the fading remains constant during the dwell time of N chips and there is no frequency offset, $\mathbf{R}_{\boldsymbol{s}}$ can be simplified to $\mathbf{1}_{N\times N}$ and, thus, $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ becomes $\Lambda_{\mathrm{C}}(\boldsymbol{x})$, from which we find that, as well as $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$, $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ also

includes $\Lambda_{\mathrm{C}}(\boldsymbol{x})$ as a special case. It should be noted that $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$ and $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ need only the statistical description of the fading process, and not the actual realizations of the fading process, in compensating for the fast fading effect before combining the correlation samples. Such a requirement, however, may limit their implementation. Thus, in the next section, suboptimal detection algorithms are discussed, which obviate the need for any information on the fading statistics (and the frequency offset).

4 Suboptimal Detection Algorithms

4.1 Chip-Level Noncoherent (CLN) Detection Algorithm

From (4) and (5), we can observe that $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$ and $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ form a *weighted* sum of $\{x_k^* x_l\}_{k,l=1}^N$ through $(\boldsymbol{\Gamma}_H^{-1} - \boldsymbol{\Gamma}_K^{-1})$ and $\mathbf{R}_{\boldsymbol{s}}$, respectively. $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$, for example, can be rewritten as

$$\Lambda_{\mathrm{CLLO}}(\boldsymbol{x}) = \sum_{k=1}^{N} |x_k|^2 + \sum_{k=1}^{N} \sum_{l=1, l\neq k}^{N} \phi\left(|k-l|\, T_c\right) e^{j2\pi\epsilon(k-l)} x_k^* x_l. \tag{6}$$

In (6), it can be seen that $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ compensates for the combined effects of fading and frequency offset on $\{x_k^* x_l\}_{k,l=1}^N$ through the weighting factor $\phi\left(|k-l|\, T_c\right)$ $\cdot e^{j2\pi\epsilon(k-l)}$, and it is also observed that the components $\left\{|x_k|^2\right\}_{k=1}^N$ are added with equal weights regardless of fading time variation and frequency offset. From this observation, we propose to use the following algorithm as a suboptimal detection algorithm, which does not require any channel information:

$$\Lambda_{\mathrm{CLN}}(\boldsymbol{x}) = \sum_{k=1}^{N} |x_k|^2, \tag{7}$$

where CLN is an abbreviation for "chip-level noncoherent", which originates from the fact that $|x_k|^2$ can be considered as noncoherent processing performed at the chip level. In fact, $\Lambda_{\mathrm{CLN}}(\boldsymbol{x})$ becomes optimal and locally optimal for a fast fading channel such that all of the elements of $\boldsymbol{x}$ are uncorrelated: specifically, in such an environment, $(\boldsymbol{\Gamma}_H^{-1} - \boldsymbol{\Gamma}_K^{-1})$ and $\mathbf{R}_{\boldsymbol{s}}$ can be simplified to $\frac{\gamma^2 \sigma_w^{-2}}{\sigma_w^2 + \gamma^2} \mathbf{I}_N$ and $\mathbf{I}_N$, respectively, and thus $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$ and $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ become $\frac{\gamma^2 \sigma_w^{-2}}{\sigma_w^2 + \gamma^2} \boldsymbol{x}^{\mathcal{H}} \mathbf{I}_N \boldsymbol{x}$ and $\boldsymbol{x}^{\mathcal{H}} \mathbf{I}_N \boldsymbol{x}$, respectively, which are equivalent to $\Lambda_{\mathrm{CLN}}(\boldsymbol{x})$ given in (7).

4.2 Chip-Level Differential (CLD) Detection Algorithm

Now, it should be observed that the second term of (6) can be considered as the sum of the components differentially processed at the chip level, with weights that depend on channel information. Using (2), the signal components of $\{x_k^* x_l\}_{k,l=1,k\neq l}^N$ can be expressed as $\left\{\gamma^2 \alpha_k^* \alpha_l e^{-j2\pi\epsilon(k-l)}\right\}_{k,l=1,k\neq l}^N$. If the phase

fluctuation due to fading between two successive correlation samples is not very significant, the signal components $\left\{\gamma^2\alpha_k^*\alpha_{k+1}e^{j2\pi\epsilon}\right\}_{k=1}^{N-1}$ (corresponding to $\left\{x_k^*x_{k+1}\right\}_{k=1}^{N-1}$) from among $\left\{\gamma^2\alpha_k^*\alpha_l e^{-j2\pi\epsilon(k-l)}\right\}_{k,l=1,k\neq l}^{N}$ would be approximately phase aligned. Moreover, they are of equal average strength. As a result, just as was done in the case of $\left\{|x_k|^2\right\}_{k=1}^{N}$, the components $\left\{x_k^*x_{k+1}\right\}_{k=1}^{N-1}$ can be added with equal weights to form an algorithm: yet, it should be pointed out that the signal component of $\sum_{k=1}^{N-1} x_k^*x_{k+1}$ is divided into real and imaginary parts, in contrast to that of $\sum_{k=1}^{N}|x_k|^2$. Hence, we take the envelope of $\sum_{k=1}^{N-1} x_k^*x_{k+1}$ to combine the divided signal parts, and thus obtain a suboptimal detection algorithm such that

$$\Lambda_{\mathrm{CLD}}(\boldsymbol{x}) = \left|\sum_{k=1}^{N-1} x_k^*x_{k+1}\right|, \tag{8}$$

where CLD is an abbreviation for "chip-level differential". $\Lambda_{\mathrm{CLD}}(\boldsymbol{x})$ is expected to be more sensitive to the fading rate than $\Lambda_{\mathrm{CLN}}(\boldsymbol{x})$, since the degree of phase coherence among the signal components of $\left\{x_k^*x_{k+1}\right\}_{k=1}^{N-1}$ depends on the fading rate.

5 Simulation Results and Discussion

We compare the detection performance of the conventional and proposed detection algorithms. In evaluating the performance, we consider the following parameters: the PN code of $L = 1023$ chips, the dwell time length $N = 256$ chips, and the false alarm probability $P_F = 10^{-2}$. The SNR/chip is defined as PT_c/N_0. The autocorrelation function, $\phi(\Delta t)$, of the fading process is taken as $\rho^{\Delta t/T_c}$ [2], [3], where $0 \leq \rho \leq 1$ is the parameter that characterizes the fading rate, such that the smaller the value of ρ, the faster the fading.

Fig. 2 shows the detection probabilities of the conventional and proposed detection algorithms for $\rho = 0.97$ and 0.95 in the absence of frequency offset. As expected, the performance of the conventional algorithm degrades substantially as ρ becomes smaller, i.e., the fading becomes faster, whereas the performance of the proposed algorithms improves as the fading becomes faster; however, an opposite trend occurs at relatively low SNR/chip values. This can be explained as follows. As the fading becomes faster, the chip-level processed components become more uncorrelated, and thus the diversity gain obtained through the combining of the components increases, resulting in better detection performance. However, the increased fading rate enhances the phase misalignments among the differentially processed components and makes the noncoherent combining loss [4] for the noncoherently processed components more pronounced. Eventually, at low SNR/chip values, such effects become more significant than the diversity effect, resulting in worse detection performance.

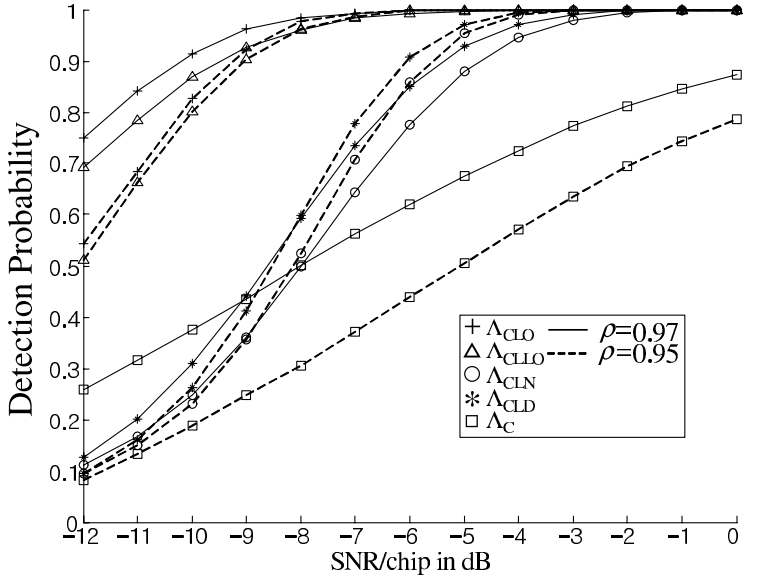


Fig. 2. Detection probability of the conventional and proposed detection algorithms for $\rho = 0.97$ and 0.95 when $\epsilon = 0$

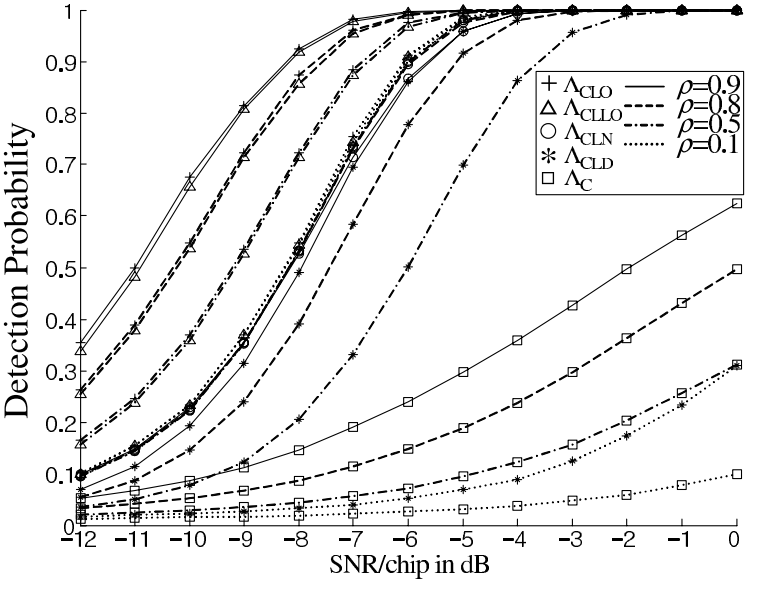


Fig. 3. Detection probability of the conventional and proposed detection algorithms for $\rho = 0.9$, 0.8, 0.5, and 0.1 with $\epsilon = 0.001$

Fig. 3 shows the detection probabilities of the conventional and proposed detection algorithms for $\rho = 0.9$, 0.8, 0.5, and 0.1 with $\epsilon = 0.001$. From this figure, we can observe that the performance of the conventional algorithm degrades severely due to fading and frequency offset. Unlike in Fig. 2, as the fading becomes faster, the performances of $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$, $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$, and $\Lambda_{\mathrm{CLD}}(\boldsymbol{x})$ are observed to degrade for the whole range of SNR/chip values shown. This is due to the fact that, as the fading becomes faster, the extent of the phase misalignments among the differentially processed components increases and its effect becomes predominant over the diversity gain effect regardless of the SNR/chip value. On the other hand, the performance of $\Lambda_{\mathrm{CLN}}(\boldsymbol{x})$ follows the same trend as that shown in Fig. 2. Finally, the performance of $\Lambda_{\mathrm{CLN}}(\boldsymbol{x})$ is found to be quite robust to variations in the value of ρ and to approach that of $\Lambda_{\mathrm{CLO}}(\boldsymbol{x})$ and $\Lambda_{\mathrm{CLLO}}(\boldsymbol{x})$ as the fading becomes faster, as stated in Subsection 4.1.

6 Conclusion

In this paper, various detection algorithms were proposed based on chip-level processing for DS/CDMA code acquisition in fast fading channels, wherein the fading process changes rapidly within the dwell time. First, we derived the joint pdf of the correlation samples obtained on a chip-by-chip basis during the dwell time. Based on this pdf and on the ML and LO detection criteria, chip-level optimal and chip-level LO detection algorithms were proposed, which require the statistics, but do not need the realization of the fading parameters. Both algorithms were found to include the conventional algorithm as a special case, wherein the fading process remains constant during the dwell time and there is no frequency offset. Two suboptimal detection algorithms were also derived. The proposed detection algorithms were shown to dramatically outperform the conventional detection algorithm as the fading becomes faster.

References

1. T.K. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-palma, "A survey of various propagation models for mobile communication," *IEEE Anten. Propag. Mag.*, vol. 45, pp. 51-82, June 2003.
2. E.A. Sourour and S.C. Gupta, "Direct-sequence spread-spectrum parallel acquisition in a fading mobile channel," *IEEE Trans. Comm.*, vol. 38, pp. 992-998, July 1990.
3. W.H. Ryu, M.K. Park, and S.K. Oh, "Code acquisition schemes using antenna arrays for DS-SS systems and their performance in spatially correlated fading channels," *IEEE Trans. Comm.*, vol. 50, pp. 1337-1347, Aug. 2002.
4. J.G. Proakis, *Digital Communications*, NY: McGraw-Hill, 2001.
5. G.H. Golub and C.F. van Loan, *Matrix Computations*, MD: Johns Hopkins University Press, 1996.
6. S.A. Kassam, *Signal Detection in Non-Gaussian Noise*, NY: Springer-Verlag, 1987.

Clustering-Based Distributed Precomputation for Quality-of-Service Routing*

Yong Cui and Jianping Wu

Department of Computer Science, Tsinghua University,
Beijing, P.R.China, 100084
cy@csnet1.cs.tsinghua.edu.cn, jianping@cernet.edu.cn

Abstract. As a potential solution to provide quality of service (QoS) for next-generation IP networks, QoS routing (QoSR) seeks to find a multi-constrained path, where the scalability and routing performance are still open problems. We propose a novel Clustering-based Distributed Precomputation algorithm (CDP) for multi-constrained QoSR. After dominating path selection is analyzed to omitting numerous dominated paths, a clustering technique is further presented for dominating path aggregation in routing computation. These two techniques in turn achieve efficient aggregation of the QoS routing table. CDP greatly decreases the computational complexity on a single node by utilizing the distributed computation on each node in the network. Simulation results confirm that CDP not only has low computational complexity, but also achieves high routing performance with good scalability on both QoS parameters and the network scale.

1 Introduction

The next-generation Internet based on IP networks is expected to support applications with diverse quality-of-service (QoS) requirements [1][2]. As an important method to provide QoS, QoS routing (QoSR) seeks to find a feasible path satisfying multiple constraints for each QoS application, and thus performs QoS control on the level of path selection from numerous paths in the network [3][4][5][6][7][8]. Since the bandwidth is heavily limited and transmission delay is relatively large in wireless networks, QoSR becomes an important potential solution for QoS control in the next-generation wireless networks.

This paper proposes a novel solution to the general QoSR problem with diverse QoS parameters: Clustering-based Distributed Precomputation (CDP). In CDP, each node, maintaining a QoS routing table, broadcasts its routing table to all of its neighbors, while receiving the routing information sent by its neighbors. In order to reduce the QoS routing table, we introduce dominating path selection and propose the

* Supported by: (1) the National Natural Science Foundation of China (No. 60403035); (2) the National Major Basic Research Program of China (No. 2003CB314801).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 551 – 558, 2005.
© Springer-Verlag Berlin Heidelberg 2005

clustering-based aggregation of dominating paths, which achieve high routing performance for CDP with low computational complexity and good scalability.

The rest of this paper is organized as follows. The problem formulation is given in Section II. We analyze the dominating paths and propose the algorithm in Section III. In Section IV, CDP is evaluated by extensive simulations. Finally, conclusions appear in Section V.

2 Problem Formulation

A directed graph $G(V,E)$ presents a network. V is the node set and the element $v \in V$ is called a node representing a router in the network. E is the set of edges representing links that connect the routers. The element $e_{ij} \in E$ represents the edge $e = v_i \to v_j$ in G. In QoSR, each link has a group of independent weights $(w_0(e), w_1(e), \Lambda, w_{k-1}(e))$, which is also called QoS weight $w(e)$, where $w_l(e) \in \Re^+$ for $l = 0,1,\mathrm{L}, k-1$. QoS weights can be divided into three classes: additive (e.g. cost, delay), multiplicative (e.g. loss rate) and concave (e.g. available bandwidth) [3].

In this paper $w_l(e)$ ($l = 0,1,\mathrm{L}, k-1$) can be any kind of QoS parameters. Since multiplicative parameters can be transformed into additive constraints, we only consider additive and concave constraints. Accordingly, for a path $p = v_0 \to v_1 \to \Lambda \to v_j$, $w_l(e) \in R^+$ and $0 \le l \le k-1$, the path weight $w_l(p) = \sum_{i=1}^{j} w_l(v_{i-1} \to v_i)$ if $w_l(e)$ satisfies the additive characteristic, or $w_l(p) = \prod_{i=1}^{j} w_l(v_{i-1} \to v_i)$ if $w_l(e)$ is multiplicative, or $w_l(p) = \max_{i=1}^{j} w_l(v_{i-1} \to v_i)$ if $w_l(e)$ is concave.

Definition 1. *Feasible path*
For a given graph $G(V,E)$, source node *s*, destination node *t*, $k \ge 2$ and a constraint vector $c = (c_0, c_1, \Lambda, c_{k-1})$, the path p from s to t is called a feasible path, if $w_l(p) \le c_l$ for any $l = 0,1,\Lambda, k-1$ (if $w_l(p)$ represents the available bandwidth of path *p*, it should be $w_l(p) \ge c$). We write $w(p) \le c$ in brief. □

Note: $w(e)$ and c are both *k*-dimensional vectors. For a given QoS request with its constraint *c*, QoSR seeks to find a feasible path *p* satisfying $w(p) \le c$ based on the network state information.

In addition to the traditional destination and the next hop, QoS routing tables need to maintain the QoS weight $w(p)$ of each path. When a QoS flow (packet) arrives at a node, the node only seeks to find a feasible path in the table and forwards the flow (packet) to the next hop accordingly.

Definition 2. *Extended distance vector*
For a given path *p* from source *s* to destination *t*, (*s*,*t*,*w*(*p*)) is called the extended distance vector of path *p*. □

Each node in the network converts the items in the routing table it maintains into extended distance vectors, and then sends them to its neighbors. Based on these vectors received by each node, a node computes its routing table with CDP.

3 Dominating Path Aggregation

Since there are different paths between two nodes in an ordinary network, a lot of routes may exist for a certain destination in the QoS routing table. Multi-constrained QoSR is an NPC problem [9] [5], so the number of routes may increases exponentially with the network scale. In order to increase the scalability of the QoSR, it is necessary to restrict the number of routes to a certain destination. Tome recent research results show that a few distinctive routes can represent the numerous paths and satisfy most QoS requests [2] [10].

3.1 Dominating Path

Path set $P_{<s,t>}$ denotes a set of paths with the same pair of source and destination <*s*,*t*>, i.e. $p \in P_{<s,t>}$ is a path from *s* to *t*.

Definition 3. *Dominating path*
For a given none-empty path set $P_{<s,t>}$ and $p \in P_{<s,t>}$, if there is no path $p' \in P_{<s,t>}$ satisfying $w(p') < w(p)$, path *p* is call a dominating path on $P_{<s,t>}$. □

The "dominating" relationship between the paths in path set $P_{<s,t>}$ is a partial order, resulting in that $P_{<s,t>}$ is a partially ordered set. Therefore, multiple minimum elements (dominating paths) may exist in $P_{<s,t>}$.

Theorem 1. Routing performance will not be decreased by omitting none-dominating paths from none-empty path set $P_{<s,t>}$.

Proof: For a none-dominating path $p \in P_{<s,t>}$, there must be a dominating path $p' \in P_{<s,t>}$ with $w(p') < w(p)$. For any QoS request from *s* to *t* with constraint *c*, if *p* is a feasible path, i.e. $w(p) \le c$, then $w(p') < c$, i.e. p' is also a feasible path for *c*. After the omission of *p* from $P_{<s,t>}$, $P_{<s,t>}$ still has the element $p' \in P_{<s,t>}$ that satisfies the request. Thus, the omission does not decrease the routing performance. □

Therefore, each node in the network may omit a lot of none-dominating paths from P<s,t> in the distributed routing process, and only maintains dominating paths for routing computation and communication.

Definition 4. *Dominating path set*
For a given none-empty path set $P_{<s,t>}$, if $\forall\ p \in P_{<s,t>}$ is a dominating path on $P_{<s,t>}$, $P_{<s,t>}$ is called a dominating path set. □

Definition 5. *Maximum dominating path set*
For a given none-empty path set $P_{<s,t>}$ and a dominating path set $A \subset P_{<s,t>}$, if $\forall p \in P_{<s,t>}\text{-}A,\ \exists p' \in A$ that $w(p') < w(p)$. *A* is called the maximum dominating path set on $P_{<s,t>}$, where *A* is denoted by $D_{<s,t>}$. □

Since the maximum dominating path set $D_{<s,t>}$ is independent to the method by which $D_{<s,t>}$ is computed or selected from $P_{<s,t>}$, we omit the detailed method to calculate $D_{<s,t>}$ in the paper.

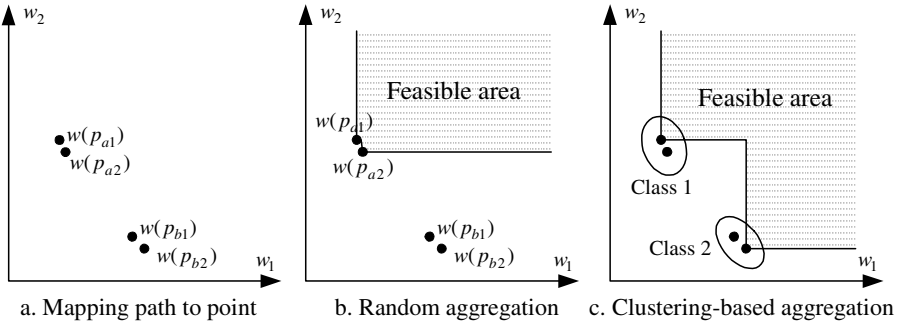


Fig. 1. Aggregation of dominating paths (R=2)

3.2 Clustering-Based Dominating Path Aggregation

Ordinarily speaking, the selection of dominating path can eliminate a lot of paths to improve the scalability without reducing the routing performance. However, in some large-scale networks, dominating paths for a certain <s,t> may still be numerous, or even be exponential to the network scale [11]. In order to improve the scalability, it needs to restrict the maximum number of dominating paths for a certain <s,t>.

Definition 6. *Maximum reduplication R*
The maximum number of dominating paths for a given <s,t> is called the maximum reduplication, denoted as R. □

According to the maximum reduplication R, each node needs to guarantee that the routes to a certain destination should be no more than R. We then analyze how to select R representative paths to satisfy QoS requests as much as possible in the QoS weight space. In a network with k QoS weights, the weight $w(p)$ of path p can be regarded as a point (or vector) in the k-dimensional Euclidean Space. Thus, the maximum dominating path set $D_{<s,t>}$ represents a point set in the k-dimensional Euclidean space.

As an example shown in Fig. 1.a, the set $D_{<s,t>}$ includes 4 dominating paths in the QoS weight space with k=2. We choose R=2, i.e. we need to select 2 paths into the QoS routing table from the 4 paths to improve the scalability. One possible method is random selection, where path p_{a1} and p_{a2} may be selected as shown in Fig. 1.b. For any QoS request with constraint c, if a selected path p satisfies $w(p) \le c$, p can be taken as the feasible path for the request. Therefore, the shadowed area in the figure represents the feasible area for requests. In order to select a more representative path from $D_{<s,t>}$ to enlarge the feasible area, another possible method is to aggregate dominating paths to R classes by clustering, and then select one path from each class. Fig. 1.c shows a possible result of the clustering-based aggregation, where p_{a1} and p_{a2} aggregate to class 1 while p_{b1} and p_{b2} compose class 2. Thus, a path is then selected from each class to

construct the feasible area. The clustering-based aggregation is generally easier to satisfy a QoS request than the random process.

```
Aggregation_Program (D<s,t>, R, T)
1)  times=0  /* iteration times */
2)  AR(t)=?  /* aggregated paths */
3)  Select R paths to PR randomly
4)  D<selected> = w(p) of R paths in PR
5)  Label R points in D<selected>
6)  DO
7)    FOR EACH path p in D<s,t>
8)      Find nearest point q in D<selected>
9)      Label p as q's label
10)   FOR EACH label
11)     IF path exists for label in D<s,t>
12)       q = average w(p) of p in D<s,t> with label
13)       Replace original point by q in D<selected>
14)     ELSE /* i.e. path does NOT exist for label */
15)       find the w(p) farthest to the point with label
16)       Replace original point by w(p) in D<selected>
17)   times=times+1
18) WHILE times<T AND D<selected> changed
19) FOR EACH point in D<selected>
20)   Add nearest path p in D<s,t> to AR(t)
21) RETURN AR(t)
```

The above Aggregation_Program shows the pseudo-code for clustering-based aggregation of dominating paths. The input of the algorithm includes the dominating path set $D_{<s,t>}$ computed by node s with the destination t, the maximum reduplication R and the maximum number of iterations T for clustering. P_R is the initial path set with R paths randomly selected from $D_{<s,t>}$. $D_{<\text{selected}>}$ is a set of R points in the Euclidean Space rather than a set of paths, where each point represents a clustered class. The output of the algorithm $A_R(t)$ is R paths to destination t after clustering, i.e. the routes from s to t.

The algorithm initializes the maximum iteration number, the path set A_R, the dependency of routing table to dominating paths, $D_{<\text{selected}>}$ as the random R paths from $D_{<s,t>}$ and the label of each point in $D_{<\text{selected}>}$ where $D_{<\text{selected}>}$ is a point set in the Euclidean Space rather than a path set. (Line 1-5). It then performs the iteration of labeling procedure, which includes the following 3 steps: (1) Labeling each path in $D_{<s,t>}$ (Line 7-9): for each path in $D_{<s,t>}$, find in $D_{<\text{selected}>}$ the point nearest in Euclidean distance to the path and label the path as the label of the point. (2) Generating the new point set $D_{<\text{selected}>}$ (Line 10-16): for each label, compute the average QoS weight of all of the paths with the label, and replace the original point with the label in $D_{<\text{selected}>}$ by this average weight (Line 11-13). If no paths exist for a certain label, the point is then replace by the weight of the farthest path to the original point with the label (Line 14-16). (3) The labeling procedure iterates until the maximum iteration number reaches or $D_{<\text{selected}>}$ is not changed in the iteration (Line 17-18). After the above iteration procedure, for each point in $D_{<\text{selected}>}$, the algorithm finds the path in $D_{<s,t>}$ nearest in Euclidean distance to the point. These paths compose the routes from s to t.

4 Performance Evaluation

Simulation results in this section show the performance evaluations of the following different algorithms. (1) CDP (i.e. the proposal in the paper). (2) CDP without clustering-based aggregation, i.e. to randomly select R paths from the dominating path set as the routes. This algorithm can be taken as the distributed extension to the path-limited heuristic proposed by Yuan, where Yuan's heuristic is a centralized algorithm [11], this algorithm is denoted as D-Yuan. The comparison between CDP and D-Yuan will show the efficiency of clustering-based aggregation. (3) CDP without both clustering aggregation and dominating path selection, i.e. to randomly select R route from the extended distance vectors maintained by a node, denoted as DPRnd. This comparison between D-Yuan and DPRnd will show the efficiency of dominating path selection. (4) MEFPA, a well performed centralized precomputation algorithm in literatures [2]. We set the number of weight divisions b=5 and $R \le C_{b+k-2}^{k-1}$ in MEFPA, and keep the number of routes consistent in the four algorithms in each simulation.

We propose the spindle network topology, as shown in Fig. 2, for the comparison of different routing algorithms. Src is the source node, where the final routing tables are compared in different algorithms. In the network, a single link exists between Src and each NB node, while multiple duplicate links exists between each NB and Dst. The number of links between an NB and Dst is denoted *DupNum*, and *DupNum* links represent that there are numerous different paths from NB to Dst.

In the simulations, we set (1) degree=10 for Src, i.e. 10 neighbors, (2) DupNum = 40, 200, 1000, 5000, respectively, (3) QoS weight number k = 2, 3, 4, 5, respectively, (4) $w_l(e) \sim$ uniform[1,1000] for $l = 0,1,\Lambda\ ,k-1$, where $w_l(e)$ have no correlation for different e or l. (5) QoS request from Src to Dst with the constraint c. Since the average weight of path p from Src to Dst is $w_l(p)$ =1000 for $l = 0,1,\Lambda\ ,k-1$, we select c_l obeying Gaussian probability distribution with mean $\mu = 1000$ and deviation $\sigma = 1000/3$. The routing performance is evaluated by the routing success ratio [2][6][7]. Computational complexity is then evaluated by running time of algorithms.

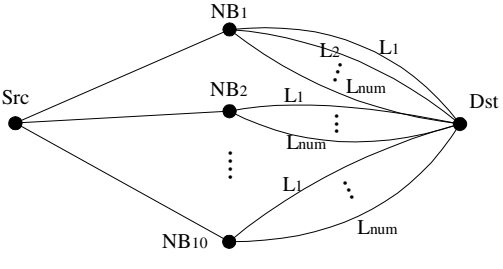
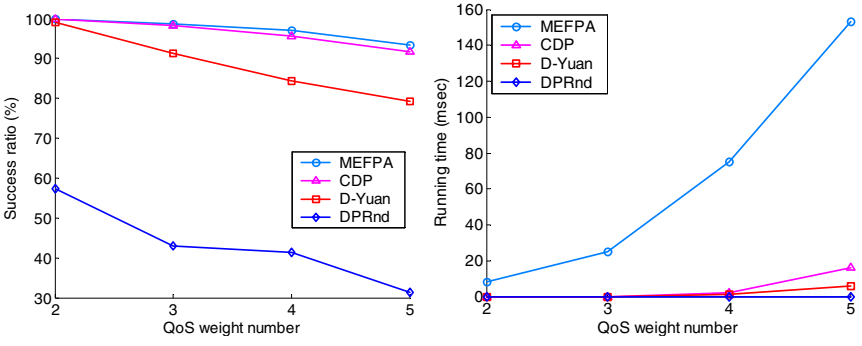


Fig. 2. Spindle network topology



a. routing performance comparison b. running time comparison

Fig. 3. Performance evaluation of four algorithms with different weight number

Fig.3 show the performance of these four algorithms with *DupNum*=1000 for large-scale networks and weight number k=2,3,4,5. The success ratio in Fig. 3a shows DPRnd < D-Yuan < CDP ≈ MEFPA in routing performance, which confirms that both dominating path selection (D-Yuan) and clustering-based aggregation (CDP) improve the routing performance effectively. Considering the QoS weight number, DPRnd and D-Yuan degrade their performance heavily with increasing k, while MEFPA and CDP sustain their performance. Although centralized MEFPA achieves a high performance by making overall plans and taking all factors into consideration on Src, the routing performance of distributed CDP is still comparable with that of MEFPA.

From the running time comparison shown in Fig. 3b, the distributed algorithms, DPRnd, D-Yuan and CDP, greatly excel the centralized MEFPA, especially in large-scale networks. In fact, the computation of centralized MEFPA is based on the whole network state information, while the input information is only $O(dR)$ for a given destination in distributed algorithms, e.g. CDP. In a real large network with complex topology, MEFPA may need more computational resources to create a shortest path tree, while CDP is independent of the network topology. Although MEFPA compute the whole routing table every time, the frequent changes of network state overload each node in the large-scale QoS network. If a long hold timer is used to reduce the computation frequency, a large staleness may lower the routing performance [12]. Therefore, considering both routing performance and running time shown in Fig.3, the advantages are apparent to deploy CDP rather than MEFPA in real IP networks.

5 Conclusion

QoSR algorithms in literature often have some limitations including connection orientation, centralized computation and limited types of QoS weights. In this paper, we introduced the extended distance vectors to support diverse QoS parameters and proposed a clustering-based distributed precomputation QoSR algorithm, namely CDP. We analyzed selection of dominating paths in the QoS weight space and gave the algorithm for clustering-based dominating path aggregation. Since we proved that

omitting none-dominating paths does not decrease the routing performance, the dominating path selection performs a lossless reduction of routes. Clustering-based dominating paths aggregation further reduces the number of routes with representative paths. Hence, both the dominating path selection and the clustering-based aggregation improve the routing performance and scalability effectively. Moreover, CDP adopts distributed computation, which radically reduces the computational complexity. Simulation results further confirm that CDP performs well in terms of both routing performance and scalability.

References

1. X. Xiao and L. M. Ni, Internet QoS: A big picture, *IEEE Network*, vol. 13, no. 2, pp. 8-18, March-April 1999.
2. Y. Cui, K. Xu, J.P. Wu. Precomputation for Multi-constrained QoS Routing in High-speed Networks. In *Proceedings of the IEEE INFOCOM 2003*, San Francisco, CA, April 2003.
3. Y. Cui, J. P. Wu, K. Xu, et al. Research on internetwork QoS routing algorithms: a survey. *Chinese Journal of Software*, vol. 13, no. 11, 2065-2076, 2002.
4. S. Chen and K. Nahrstedt, An overview of quality-of-service routing for next-generation high-speed networks: Problems and solutions, *IEEE Network*, vol. 12, no. 6, pp. 64-79, Nov. 1998.
5. Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, Sep. 1996.
6. Yong *Cui*, Ke Xu, Jianping Wu, and et al. Multi-constrained routing based on simulated annealing. *IEEE ICC* 2003.
7. T. Korkmaz, M. Krunz, Multi-Constrained Optimal Path Selection. *IEEE INFOCOM'01*, vol.2, pp. 834 -843, 2001.
8. L. Xiao, K. S. Lui, J. Wang and K. Nahrstedt. QoS Extension to BGP. *IEEE ICNP*, 2003.
9. M. S. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, New York, 1979.
10. A. Orda and A. Sprintson, QoS routing: the precomputation perspective. *IEEE INFOCOM'00*, vol. 1, pp. 128-136, 2000.
11. X. Yuan, X. Liu, Heuristic Algorithms for Multi-Constrained Quality of Service Routing, *IEEE INFOCOM'01*, vol.2, pp. 844 -853, 2001.
12. A. Shaikh, J. Rexford, K. G. Shin, Evaluating the impact of stale link state on quality-of-service routing. *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, Apr. 2001

Traffic Grooming Algorithm Using Shortest EDPs Table in WDM Mesh Networks*

Seungsoo Lee, Tae-Jin Lee, Min Young Chung, and Hyunseung Choo

Lambda Networking Center,
School of Information and Communication Engineering,
Sungkyunkwan University,
440-746, Suwon, Korea
{isoolee, tjlee, mychung, choo}@ece.skku.ac.kr

Abstract. In optical networks with huge transmission capability, Wavelength Division Multiplexing (WDM) has been actively studied in the research community. Traffic grooming technology based on divided bandwidth of a wavelength by WDM is very important for the network cost. This paper proposes a traffic grooming algorithm that employs the table for shortest Edge Disjoint Paths (EDPs) with clever selection on demands. Comprehensive simulations in various network environments show that the proposed algorithm outperforms well-known Maximizing Resource Utilization (MRU) up to 15% for the network throughput and up to 17% for the running time.

1 Introduction

Due to the explosive growth of hosts, users and services for the Internet, high speed network technologies based on optical networks have been appearing. Nowadays, Internet data traffic is increasing rapidly by more than 100% every year and in actuality, data traffic is passing existent voice traffic. Therefore existing networks cannot correspond with next generation data demand and capacity. Hence, in this context, research on optical technologies is drawing more interest for future high-speed network infrastructures [1].

Routing and wavelength assignment (RWA) for WDM networks, which are based on wavelength granularity, has been generally investigated [2]-[4]. As WDM technology continues to mature, there exists a bandwidth difference between the transmission capacity (e.g., OC-48, OC-192, OC-768) and typical connection requests with low-late traffic stream (e.g., OC-1, OC-3, OC-12, etc.) [5]. If the entire bandwidth of a wavelength channel is allocated to a low-late traffic stream, a large portion of the transmission capacity might be wasted. In order to effectively use network resources, many low-rate traffic streams should be efficiently groomed or multiplexed into a ligthpath [6].

* This work was supported in parts by Brain Korea 21 and the Ministry of Information and Communication, Korea. Corresponding author: Prof. H. Choo.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 559–567, 2005.
© Springer-Verlag Berlin Heidelberg 2005

This paper deals with the traffic grooming problem in WDM mesh networks with limited number of wavelengths for the environment of static traffic requests. Our objective is to maximize the network throughput. Since the optical network resource is limited, to maximize a throughput by increasing utilization rate of wavelengths is very important in terms of cost. The proposed algorithm works as follows. For a given network, it finds the shortest Edge Disjoint Paths (EDPs) [2] for all source and destination pairs and saves them to the shortest EDPs table (EDPT). It then calculates a resource utilization rate per hop for each request. Finally, it chooses a request with the biggest value of resource utilization rate, and sets the path corresponding to the selected request based on the information in EDPT. From simulation results, network throughput of the proposed algorithm is increasing up to about 10% in NSFNET and up to about 15% in random networks compared with that of the previous MRU algorithm. In operation time, the proposed algorithm is about 17% faster than the previous MRU in NSFNET.

The rest of the paper is organized as follows. In section 2, previous works on WDM SONET ring and mesh WDM networks, single-hop grooming and multi-hop grooming, and routing method for traffic grooming are presented. In section 3, we propose a traffic grooming algorithm in WDM mesh networks and describe the algorithm with an illustrative example. In section 4, the performance evaluations are presented and discussed. Finally, section 5 concludes the paper.

2 Related Works

2.1 Previous Works in Various Environments

Recently the type of optical transport network topologies is evolving from interconnected SONET/WDM ring to optical WDM mesh. The traffic grooming problem in optical WDM mesh networks has become an important area of research [6]. In [5], a generic graph model was proposed for provisioning multi-granularity connections in multi-wavelength optical WDM networks. The authors presented that for different network optimization objectives, their corresponding route selection schemes can be easily applied to the graph model. In [6], a heuristic traffic grooming algorithm is proposed for static traffic environments; maximizing resource utilization (MRU) algorithm which is known to be effective.

In general, to solve the traffic grooming problem, it is assumed that traffic requests are carried through single-hop grooming or multi-hop grooming. In single-hop grooming an end-to-end connection consists of a single lightpath. This means that only end-to-end traffic grooming is allowed. On the other hand, in multi-hop grooming a connection consists of multiple lightpaths. Thus, a connection can be dropped at intermediate nodes and groomed with other low-capacity connections on different lightpaths before it reaches its destination node. It is expected that the multi-hop grooming leads to a higher throughput than achieved by single-hop grooming.

2.2 Routing for Traffic Grooming

In general routing algorithms can be classified into three categories; fixed, adaptive, and fixed-alternate routing [5]. In fixed routing, each connection is always routed through a predefined route for a its source-destination pair. Fixed shortest path routing is an example of this approach. The shortest path for each source-destination pair is calculated offline using shortest path algorithms. If there are not enough resources to satisfy a connection request, it is blocked.

In adaptive routing, the route from a source to a destination node is chosen dynamically, as current network status. The network status depends on the connections served. For example, when a connection request arrives, the shortest path between its source and destination node is determined as the available resources in the network.

In fixed-alternate routing, multiple routes are determined when ever a connection request occurs. In this scheme, each node within the networks has a table to maintain information, including the order of multiple routes on a destination node. For example, these routes can be ordered as the first shortest path, the second shortest path, and so on. When a connection request occurs, its source node sequencelly attempts to establish a route based on the order information in the routing table, until it is successfully completed.

3 Proposed Algorithm

We propose a new heuristic algorithm for the traffic grooming problem. Define a network topology $G(N, E, W)$ for a given WDM optical network, where N is the set of nodes, E is the set of edges, W is the set of wavelengths in one fiber (each fiber is assumed to have the same number of wavelengths) and the capacity of each wavelength is C_w. Let $D=\{d_i(s, t) \mid \forall\ s, t \in N \text{ and } 1 \leq i \leq |D|\}$ denote the traffic request between same source node s and destination node t. And $C(d(s, t))$ denotes the aggregate capacity of traffic request between the node pair $d(s, t)$. Let $H(s, t)$ denote the hop distance on the physical topology between the node pair (s, t). We define $R(s, t) = C(d(s, t)) \ / \ H(s, t)$ as the connection resource utilization value, which represents the average traffic per wavelength link. This quantity shows how efficiently resources have been used to carry traffic requests.

The MRU algorithm sorts all of the node pairs (s, t) in a descending order according to network resource utilization $R(s, t)$ and put them into a list L, then attempts to assign lightpaths sequentially. But after a lightpath for one node pair is assigned, network resource utilization of the other node pairs changes since the network state on the physical topology changes. The MRU algorithm does not consider changed network resource utilization, hence this algorithm does not reflect practical network resource utilization. This method cannot be applied to the values $R(s, t)$, which were calculated at the beginning. Eventually it influences in network throughput.

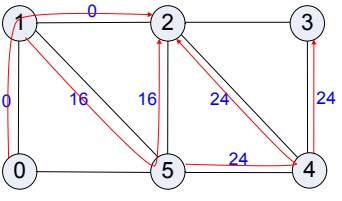
Our proposed algorithm calculates every available shortest EDPs for each $d(s,\ t)$ pair, and stores them to table A. Similar to the MRU algorithm, all

Traffic	d(s, t), C(d(s, t))	H(s, t)	R(s, t)
1	(0, 2), OC-72	2	36
2	(1, 2), OC-32	1	32
3	(1, 4), OC-60	2	30
4	(5, 2), OC-24	1	24
5	(5, 3), OC-48	2	24
6	(4, 3), OC-24	1	24
7	(2, 4), OC-12	1	12

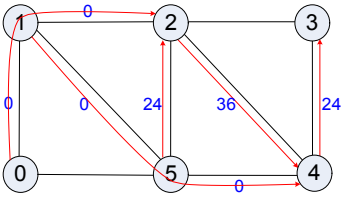
(a) Set of connection requests, L

(s, t)	Shortest EDPs
(0, 2)	0-1-2, 0-5-2
(1, 2)	1-2
(1, 4)	1-2-4, 1-5-4
(5, 2)	5-2
(5, 3)	5-2-3, 5-4-3
(4, 3)	4-3
(2, 4)	2-4

(b) Shortest EDPT



(c) MRU algorithm



(d) Proposed algorithm

Fig. 1. Example of MRU and the proposed algorithm

of the node pairs $d(s, t)$ are sorted according to network resource utilization $R(s, t)$ between (s, t) pairs and puts them into a list L in a descending order. Then it finds the node pair with the highest $R(s, t)$ and attempts to assign a lightpath for this node pair $d(s, t)$. If there is enough resources in the network in one of the paths of table A, connection request can set up a lightpath. If there is not enough resources to establish a lightpath, the proposed algorithm tries the next connection request in list L. Therefore the values $R(s, t)$, which were calculated at the beginning, will be reflected to the decision of lightpaths and it can increase the throughput. Moreover, the proposed algorithm using the fixed-alternate routing method can expect fast routing than the MRU algorithm using adaptive routing.

We present an example for the proposed algorithm and the MRU algorithm for a simple network (Fig. 1). Let us assume that it is a single fiber WDM optical network with each fiber having one wavelength channel, and the capacity of wavelength is OC-48. The aggregated traffic for node pair (0, 2), (1, 2), (1, 4), (5, 2), (5, 2), (4, 3), and (2, 4) is 72, 32, 60, 24, 48, 24, and 12, respectively. Those traffic requests are sorted by $R(s, t)$ and put into a list L (Fig. 1(a)). Then it tries to setup a lightpath for the first node pair $d(s, t)$ in list L. Fig. 1(c) shows the setup procedure for a lightpath for in the MRU algorithm. We can see that connection requests of 1, 2, 4, and 6 are accomodated according to the order of list L. In this case, overall network throughput is OC-128. On the other hand, our proposed algorithm tries to set up lightpaths based among the stored shortest EDPT (Fig. 1(b)). The network throughput with the proposed algorithm is OC-156. Which shows that proposed algorithm performs better than the MRU algorithm.

The proposed algorithm can be divided into two parts and it employs both fixed-alternate routing and adaptive routing. Step 1 constructs a virtual topology: using fixed-alternate routing, it tries to establish lightpaths by the resource utilization values. Step 2 conducts traffic grooming on virtual topology: using adaptive routing, it tries to find out multi-hop routes for unassigned traffic requests. The procedure and pseudo-code for the proposed algorithm are as follows.

Step 1: Construct a virtual topology:

1.1: Aggregate traffic requests for each node pair $d(s, t)$ with the same (s, t). $C(d(s, t))$ is the sum of unassigned capacity of traffic requests.
1.2: Compute all shortest EDPs and $H(s, t)$ for each node pair $d(s, t)$, and put them into a table A.
1.3: Compute $R(s, t) = C(d(s, t)) / H(s, t)$. $R(s, t)$ is the connection resource utilization value
1.4: Sort all of the node pairs $d(s, t)$ according to $R(s, t)$ and put them into a list L.
1.5: Extract the element $d'(s, t)$ with the greatest value from list L, and route according to shortest EDPs of table A. If it fails, delete $d'(s, t)$ from L, otherwise, let $C(d'(s, t)) = max\{C(d'(s, t)) - C_w, 0\}$
1.6: Go to step 1.5 until list L is empty.

Step 2: Traffic grooming on virtual topology:

2.1: Satisfy all of the connection requests that can be carried through a single lightpath hop, and update the virtual topology network state.
2.2: Route the remaining connection requests using currently available spare capacity of the virtual topology based on their connection resource utilization value $R(s, t)$. If there is no route for $d(s, t)$, then reject this connection request.

(Pseudo-code for Proposed Algorithm)

Input: Network topology $G(N, E, W)$ and demand set

$$D=\{d_i(s, t) \mid \forall\ s, t \in N \text{ and } 1 \leq i \leq |D|\}$$

Output: Total network throughput T

1. Find shortest EDPs for the demand set and store them in table A
2. Calculate $R(s, t) = C(d(s, t)) / H(s, t)$ for all demands
3. Sort all demands in descending order by $R(s, t)$ and store them in list L
4. **For** $(\lambda = 1; \lambda \leq W; \lambda{+}{+})$
5. // Construct a virtual topology
6. **While** $(L \neq \emptyset)$
7. Select $d'(s, t)$ which is on top of L
8. **If** $\exists$ a path in A for $d'(s, t)$ **then**
9. Assign the path to $d'(s, t)$ for the lightpath set up
10. $C(d'(s, t)) = max\{C(d'(s, t)) - C_w, 0\}$
11. **If** $C(d'(s, t)) = 0$ **then** delete $d'(s, t)$ from list L

12. **Else** delete $d'(s, t)$ from the list L and store to list L'
13. **End while**
14. // Routing $d(s, t)$ on the virtual topology
15. Update the network state and restore list $L = L \cup L'$
16. Route the remaining $d(s, t)$s on the current network state by multi-hop grooming
17. **If** $\lambda = W$ **then**
18. The remaining $d(s, t)$ in list L is blocked
19. **End for**
20. T = ((Total demand capacity - Blocked demand capacity) / Total demand capacity) $\times$ 100
21. Return T

In the proposed algorithm, inputs are network topology $G(N, E, W)$ and demand set D, and an output is network throughput T. After finishing the single-hop grooming, update the topology then try to setup lightpaths for the remaining requests by using multi-hop traffic grooming. The algorithm continues until there are no more wavelengths. After using all of the wavelengths, the remaining requests are blocked.

4 Performance Evaluation

In this section we evaluate the performance of the proposed algorithm by computer simulation in terms of the network throughput that is successfully carried traffic capacity and the running time. For the comparison study, we compare our proposed method with the MRU. For the evaluation in the typical network, we use NSFNET which has 14 nodes and 21 links. Moreover we employ randomly generated networks structures because we want to test those two schemes properly in various network topologies. The random connection request model proposed in [6] to generate our incoming traffic matrices is utilized here. In our case, we assume the capacity of each wavelength is OC-48, and allow the traffic bandwidth requests to be any of OC-1, OC-3, and OC-12. The traffic metrics are randomly generated that the numbers of connection requests on OC-1, OC-3, and OC-12 for all possible node pairs are generated as uniformly distributed random numbers between 0 and 16, between 0 and 8, and between 0 and 2, respectively. We simulate 1,000 times for each network topology and this is exactly same as experiments done in [6]. We obtain a random network for the physical topology by specifying the number of nodes in the graph model (N) and the probability of edge existence (P_e) for any node pair[3].

In this work, we define the network throughput as follows:

$$T = \frac{\sum_{i=1}^{|D|} C(d_i(s,t)) \times S_i}{\sum_{i=1}^{|D|} C(d_i(s,t))} \times 100\ (\%)$$

where $S_i = 1$ if connection request i has been satisfied; otherwise $S_i = 0$, and the total number of connection requests is $|D|$.

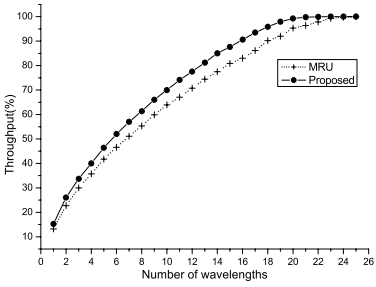
Fig. 2(a) shows the results of two algorithms compared and it demonstrates the relationship between the network throughput(T) and the number of wavelengths. When the number of wavelengths is either 1 or 2, the proposed scheme and the MRU show almost the same trends because network resources are not enough for total connection requests. However, as the number of wavelengths increases, the proposed scheme shows the better performance than the MRU. Specifically, it outperforms the MRU by 10% in terms of the network throughput when the number of wavelengths reaches to 10 through 15. As you see, when the number of wavelengths is in between 22 and 25, our scheme and the MRU both show the similar results since network resources are quite enough for total connection requests.

As mentioned earlier we perform the simulation study for random networks in order to have the generality in network structures. Simulation environments are similar to those of the fixed NSFNET except the variable P_e for the probability of edge existence. We consider P_e values as 0.3 and 0.6, and the number of nodes as 15 and 20 for the reasonable sizes of experiments. If P_e is high, we create dense networks. Otherwise, we have sparse ones.

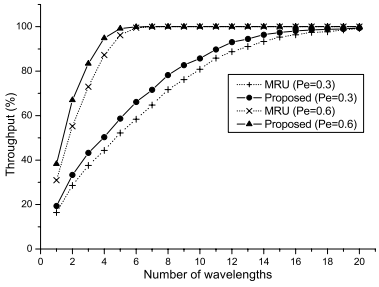
Fig. 2(b) shows that the network throughput versus the number of wavelengths for the 15-node random networks. As a whole, the proposed algorithm performs better than the MRU. When P_e is 0.3, the number of wavelengths needed per link is 20 for routing all connection requests. In the meantime, for P_e=0.6, the number of wavelengths required per link is 6 for accommodating all demands. Because, if P_e is 0.6, then the network has more edges connected to each node than the case of P_e=0.3. It means that the number of possible paths which handle the requests is increased, so they can manage requests with relatively less number of wavelengths. As a result, the proposed algorithm improves the throughput up to about 10% and 14% compared with the MRU for P_e=0.3 and P_e=0.6, respectively, in 15-node random networks.

Fig. 2(c) presents the same case for 20-node random networks. It shows quite dense networks for P_e=0.6 in 20-node random networks, and as expected there are more shortest EDPs. It provides a flexibility for identifying paths, and that is why the proposed algorithm shows the better performance than the MRU. Therefore the proposed one outperforms the MRU by 15% in terms of the network throughput for such dense networks.

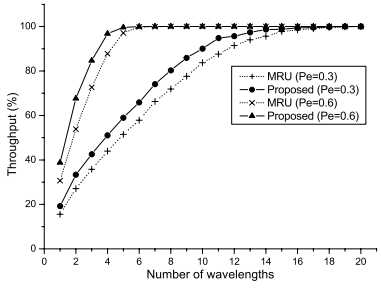
Fig. 2(d) shows that the running time for the algorithm including the selection of EDPs and wavelength assignments for requests. We generate the sets of traffic matrices for demand sets. As the number of demand sets increases, we measure the time. If there are a small number of demand sets (i.e. sets of traffic matrices), the proposed algorithm takes more time to find the shortest EDPs and to construct the table for EDPs than the MRU. However, as the number of demand sets increases, it runs faster. The table for shortest EDPs is never changed as long as the physical network structure is not changed. The proposed algorithm references the table to set up the lightpaths, meanwhile the MRU finds



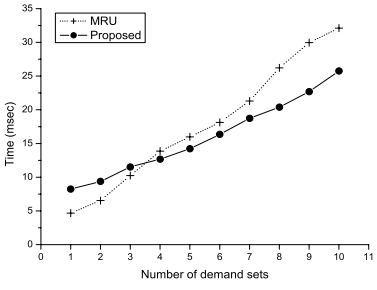
(a) Throughput in NSFNET



(b) 15-node random networks



(c) 20-node random networks



(d) Running time in NSFNET

Fig. 2. Results from the simulation study

a new shortest path in the current network state after setting up one request. If there are 10 sets of traffic matrices, it is about 17% faster than the MRU.

5 Conclusion

In this paper we study the traffic grooming problem in WDM mesh networks with the limited number of wavelengths for the environment of static traffic requests. The proposed algorithm is based on the table for shortest Edge Disjoint Paths (EDPs) with clever selections on demands. The objective of this work is to increase the network throughput and to decrease the running time as others do. Performance evaluations are done with fixed NSFNET and varied random networks. Comprehensive simulation shows that the proposed algorithm significantly improves the throughput up to 15% and the running time up to 17% compared to the previous MRU. This work will be applied to WDM multicasting networks in the near future.

References

1. H. Choo, M. Son, M.Y. Chung and T.J. Lee, "Shared Protection by Concatenated Rings in Optical WDM Networks," Springer-Verlag LNCS, vol. 3042, pp. 1476-1482, May 2004.
2. P. Manohar, D. Manjunath and R.K. Shevgaonkar, "Routing and Wavelength Assignment in Optical Network From Edge Disjoint Path Algorithms," IEEE Communications Letters, vol. 6, pp. 211-213, May 2002.
3. M.H. Kim, H. Choo, B.H. Yae, J.H. Lee, "A Practical RWA Algorithm Based on Lookup Table for Edge Disjoint Paths," Springer-Verlag LNCS, vol. 2713, pp. 138-147, June 2003.
4. H. Choo, and V.V. Shakhov, "Routing and Wavelength Assignments in Optical WDM Networks with Maximum Quantity of Edge Disjoint Paths," Springer-Verlag LNCS, vol. 3038, pp. 1138-1145, June 2004.
5. H. Zhu, H. Zang, K. Zhu, and B. Mukherjee, "A novel, generic graph model for traffic grooming in heterogeneous WDM mesh networks," IEEE/ACM Trans. Networking, vol. 11, pp. 285-299, Apr. 2003.
6. K. Zhu and B. Mukherjee, "Traffic grooming in WDM optical mesh networks," IEEE Journal on Selected Areas in Communications, vol. 20, pp. 122-133, January 2002.

Efficient Indexing of Moving Objects Using Time-Based Partitioning with R-Tree*

Youn Chul Jung, Hee Yong Youn, and Ungmo Kim

School of Information and Communications Engineering,
Sungkyunkwan University,
Suwon, Korea
{Jimmy4u, youn, umkim}@ece.skku.ac.kr

Abstract. A number of spatiotemporal access methods such as 3DR-Tree and MV3R-tree have been proposed for timestamp and interval query. These access methods consist of a single index structure covering the entire time span, and thus the performances quickly degrade since the indexed time region gets large as time progresses. To overcome the problem, we propose to employ time-based partitioning on the R-tree called Time-Based Partitioning R-tree (TPR-Tree). Since the structure of the TPR-Tree efficiently fits various queries on two dimensional data, it significantly outperforms other access methods for queries of various timestamps and time intervals. Extensive simulation validates the performance of the proposed scheme.

Keywords: Indexing moving object, location based service, R-Tree, spatiotemporal database, time stamp and interval query.

1 Introduction

Recently, due to the progress of wireless networking with GPS (Global Positioning System) and wide spread of personal communication devices such as mobile phone, concerns on contents services based on user location are increasing. The contents services are called LBS (Location Based Services). The LBS is defined as the service which tracks the location of mobile user and provides useful information associated with the location such as map service, traffic information service, traveling guide service, vehicle navigation service, public service, and so on. In order to quickly search for the requested locations for a variety of services, DBMS system needs an access method capable of efficiently searching a large amount of objects.

Most recently developed spatiotemporal access methods such as HR-Tree [1], MV3R-Tree [3], 3D R-tree [4], TB-Tree, and STR-Tree [4] are for indexing timestamp and interval queries of moving objects. The spatiotemporal access methods cover the entire time range of the network managed using a single index structure.

* This research was supported by the Ubiquitous Autonomic Computing and Network Project, 21st Century Frontier R&D Program in Korea and the Brain Korea 21 Project in 2004. Corresponding author: Hee Yong Youn.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 568 – 575, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Therefore, the performance may decrease as time passes since the size of indexed time gets very large. Also, they can neither efficiently support a variety of queries nor reflect the change of various queries along with time intervals.

To solve the problems, we focus on reducing the cost of various queries, i.e. timestamp and interval query. We propose a multiple-indexing method using time-based partitioning called time-based partitioning R-tree (TPR-Tree). It efficiently fits various queries (i.e. timestamp query and interval query) from the given spatial region of a fixed network. Compute simulation reveals that the proposed scheme outperforms other access methods for various queries, while the improvement gets substantial for timestamp query.

The rest of the paper is organized as follows. Section 2 reviews the existing access methods for timestamp and interval query. Section 3 describes the structure of the proposed TPR-Tree, and insert and search operation with the proposed structure are demonstrated. Section 4 presents the results of performance comparisons. Finally, we conclude the paper in Section 5.

2 Related Work

In this section we introduce three typical access methods – 3D R-Tree, MV3R-Tree, and HR-Tree.

2.1 3D R-Tree

The 3D R-tree [4] simply considers time as another dimension of the R-tree. Whenever an object moves to another position or changes its shape, a new minimum bounding region (MBR) is created to represent the change of the object. The MBR containing the spatial extent and lifespan is inserted in the 3D R-tree.

Figure 1 shows an example of the 3D R-tree. In this figure, R_1, R_2, and R_3 represent the MBRs describing the movements of the objects, and R_0 contains R_1, R_2, and R_3. As time passes, thus, the time region enlarges accordingly. This causes the performance of the 3D R-tree to degrade because a single R-tree keeps the whole time region in the 3D R-tree. Also, long-lived records result in a huge dead space not covering any record but being a part of the MBR. Therefore, they degrade the query performance of the 3D R-tree. However, the 3D R-tree takes smallest space since it has no duplicate data.

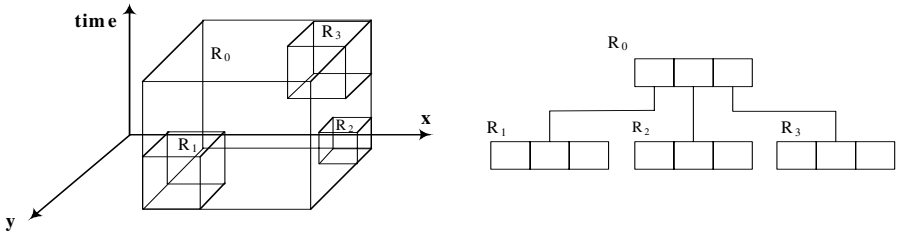


Fig. 1. The stucture of the 3D R-Tree

2.2 MV3R-Tree

The MV3R-tree [3] combines the Multi-Version R-tree (MVR-tree) and a small auxiliary 3D R-tree built at the leaf nodes of the MVR-tree. The former is used to answer timestamp and short interval queries and the latter is to answer long interval queries. Figure 2 illustrates the structure of the MV3R-tree.

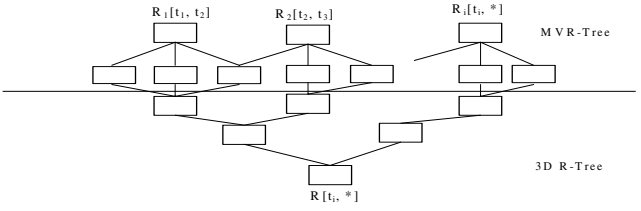


Fig. 2. The stucture of the MV3R-Tree

As shown in the figure, the MVR-tree consists of multiple R-trees that have their own jurisdiction intervals. For example, $R_1[t_1,t_2]$ deals with the records whose time intervals belong to $[t_1,t_2]$, while $R_2[t_2,t_3]$ manipulates the records whose time intervals belong to $[t_2,t_3]$. Although the size of the auxiliary 3D R-tree is very small since it shares the leaf nodes of the MVR-tree, it improves the performance of interval queries and provides flexibility to the algorithm in processing other spatial queries such as join and k-nearest neighbor.

2.3 HR-Tree

The Historical R-tree (HR-tree) [1,2] creates an R-tree whenever the objects in the previous R-tree change their positions or shapes, but common branches of consecutive R-trees are stored only once in order to save the space.

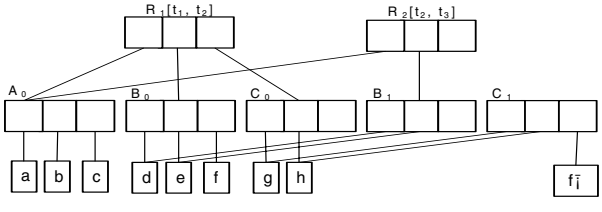


Fig. 3. The stucture of the HR-Tree

The timestamp query is directed to the corresponding R-tree and the search is performed only inside the tree. Thus, the timestamp query becomes an ordinary window query and is handled very efficiently. The interval query should search the corresponding R-trees of all the timestamps involved. Even when only one object has changed its position, the HR-tree may update the nodes contained in the path between the leaf node corresponding to the object and the root node. Therefore, the size of the HR-tree is several times larger than that of the 3D R-tree. Figure 3 shows an example

of the HR-tree. $R_1[t_1,t_2]$ deals with the records whose time intervals are contained in time interval $[t_1,t_2]$, and $R_2[t_2, t_3]$ manipulates the records whose time intervals are contained in $[t_2,t_3]$. Suppose that object f changes its position to f′ from B_0 to C_1 at $[t_2,t_3]$. The three nodes A_0, B_0, C_0 which are associated with f and f′ are copied and updated to nodes C_1.

3 The Proposed Scheme

Most spatiotemporal access methods consist of a single index structure covering the entire time span of the network, which degrades the performance as time progresses. To solve the problem, we propose the time-based partitioning R-Tree (TPR-Tree) consisting of 2-dimensional networks. We first introduce the structure of the TPR-Tree. We then describe the insertion and search algorithm based on the TPR-Tree.

3.1 The Structure of the TPR-Tree

As shown in Figure 4, the TPR-Tree consists of a single 2D R-Tree and multiple 1D R-Trees. The 2D R-Tree is used to index the spatial data of the spatial region. It is similar to the original R-Tree [9,10] and each leaf node of the 2D R-Tree contains pointers pointing the root of each 1D R-Tree on the given spatial data of the region. Consequently, each leaf node of the 2D R-tree corresponds to a 1D R-tree. Each of the 1D R-Tree is responsible for a fixed time interval derived from various queries. Also, each 1D R-Tree is generated according to the time intervals and used to index various queries on the given spatial region. Accordingly, the TPR-Tree can be considered as multiple 1D R-Trees on top of a single 2D R-Tree.

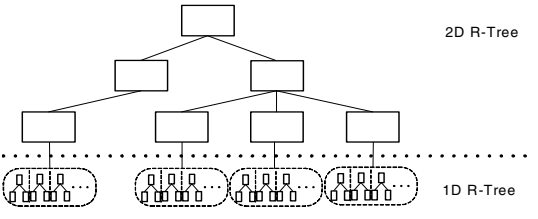


Fig. 4. The stucture of the TPR-Tree

Figure 5 illustrates the structure of the 1D R-tree. Each leaf node entry of the 1D R-Tree is of the form <ID, MBR, (t_{start}, t_{end})>. Here ID is the object identifier, MBR covers the boundaries of the children nodes, and (t_{start}, t_{end}) are the times when the record is inserted and deleted, respectively. If an entry has not yet been logically deleted, t_{end} is marked as "*". The time region boundary designates the lifespan of the record of each 1D R-Tree, and L_1, L_2, and L_i designate the interval of R_1, R_2, and R_i.

We apply time-based partitioning to the 1D R-Tree, which splits the 1D R-tree according to time interval. As a result, the TPR-Tree contains multiple 1D R-Trees in a given MBRs on a 2D R-Tree. The 1D R-Tree can dynamically support various queries and efficiently reflect the change of queries along with time interval. The

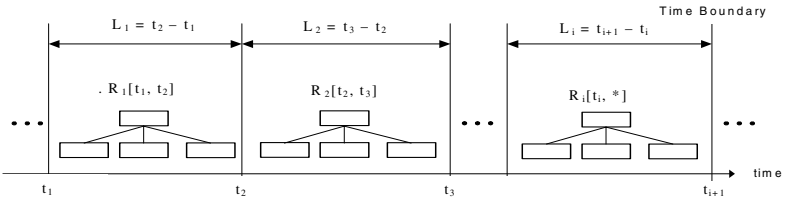


Fig. 5. The stucture of the 1D R-Tree

structure of the 1D R-tree is similar to the MV3R-tree, but the TPR-Tree does not have a shared node and employ the version spilt policy.

The processing of timestamp and interval query in the 1D R-Tree is similar to that of the 3D R-Tree [4]. However, since the 3DR-Tree is built using a single tree structure, it should search a large space for timestamp query. Note that the structure of 1D R-Tree dynamically fits various queries of time interval and timestamp. This reduces the search space and also dead space with time interval partition.

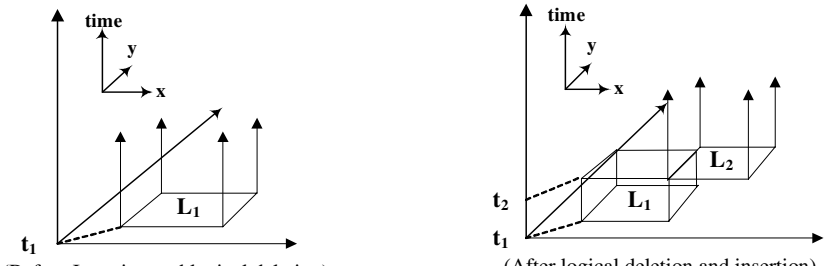


Fig. 6. The insertion and logical deletion

Figure 6 shows how insertion and deletion process work. Record R_1 is logically deleted by time boundary at timestamp t_2, where L_1 and L_2 indicate the spatial MBR of R_1 and R_2, respectively. $[t_1,t_2]$ and $[t_1,*]$ indicate the lifespan of R_1 and R_2, respectively.

3.2 Insertion and Search with the 1D R-Tree

When an insertion occurs at a specific spatial region with the TPR-Tree, executed the following steps are executed. In the 1st step, the R-Tree insertion algorithm is executed in the 2D R-Tree in order to insert the MBR covering the specific region of the 2D R-Tree leaf node. It then follows the pointer to the 1D R-Tree and finds the 1D R-trees with time interval $[T_{start},T_{end}]$. In the 2D R-Tree, we use the original R-Tree algorithm [9, 10] to index a spatial region. The next step is to execute the 1D R-tree algorithm.

Figure 7 shows the insertion algorithm with the 1D R-Tree. The algorithm of the 1D R-Tree employs the concept of time boundary. Therefore, if the lifespan of a record does not include the boundary, the record is inserted in the 1D R-Tree covering the suitable interval that contains the lifespan of the record. However, if the records inter-

sect between time intervals, each of the records split by data fragmentation is inserted in the 1D R-tree of suitable time interval containing the lifespan of the record.

```
Procedure insertion
Begin
/*Find Ri which contains the record and insert it
  into Ri*/
for Ri(ti, ti+1) /* i=1 … k */
    if (ti, ti+1) contains current time interval
        insert record to Ri;
        break;  /* exit for loop */
    endif
/* the logical deletion */
if (current record=(ID, MBR,[ti+1, *])
    previous record=[ti, *] is changed into
    previous record=[ti, ti+1];
endif
/*record intersect between time intervals*/
  for Ri(ti, ti+1) /* i=1 … K */
    if(ti, ti+1) intersect the lifespan of the record
        insert record to suitable time interval's Ri;
    endif
End
```

Fig. 7. Insertion algorithm with the 1D R-Tree

4 Performance Evaluation

This section describes the simulation environment and provides the results of simulation.

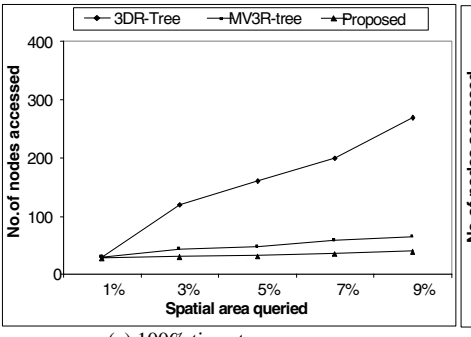
4.1 Simulation Environment

In all our experiments we use a 2.4GHz Pentium IV machine with 1GB of main memory. Due to unavailability of movement data of actual object, we use a synthetic dataset generated by GSTD methods [8] which has been widely employed as a benchmarking environment for studying the access methods handling moving points and region. It was designed to generate realistic data for the evaluation of database algorithms with indexing and storing dynamic location data.

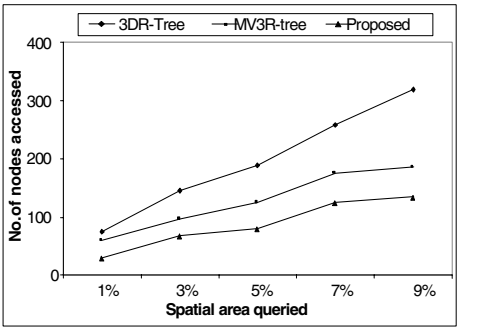
The number of moving objects is 1,000. Spatial areas queried are 1%, 3%, 5%, 7%, and 9% of the whole space. The simulation evaluates the cost of queries for combined timestamp and interval time.

4.2 Simulation Results

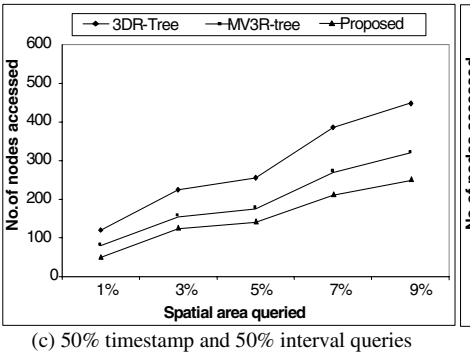
Figure 8 shows the performance of each access method for various queries of time interval and timestamp in terms of the number of nodes accessed. We compare MV3R-



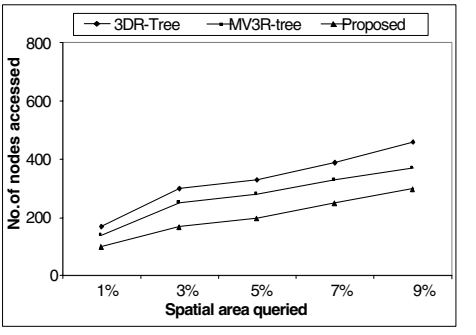
(a) 100% timestamp query



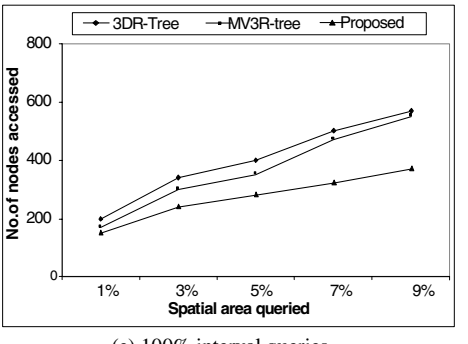
(b) 75% timestamp and 25% interval queries



(c) 50% timestamp and 50% interval queries



(d) 25% timestamp and 75% interval queries



(e) 100% interval queries

Fig. 8. The comparisons of the cost of query of the three methods

tree, 3DR-Tree, and the proposed TPR-Tree. As shown in Figure 8, the TPR-Tree outperforms the other methods for various queries of timestamp and interval time.

Notice that performance of the 3D R-tree declines with timestamp query. This is because a single R-Tree keeps the entire time region in the 3DR-Tree while the time region gets large as time passes. The proposed TPR-Tree consists of multiple indexing trees and 1D R-Tree with the time-based partitioning method. It can thus dynamically fit various queries on the given interval time.

5 Conclusion

Most existing spatiotemporal access methods consist of a single index structure covering the entire time span of the network. The performance would decrease since the indexed time of a specified region enlarges as time progresses. Also, they can neither support a variety of queries efficiently nor reflect the change of the queries with time-intervals. We have proposed a method consisting of multiple trees of two dimensions with time-based partitioning to the indexing. It can efficiently handle various queries (i.e. timestamp query, interval query) from the given spatial region. The interval length of the TPR-Tree well fits time interval of the query. Computer simulation reveals that the proposed TPR-Tree outperforms other methods for various queries of timestamp and interval time. We will further investigate the performance of the proposed approach for other operational environment.

References

[1] Nascimento, M.A., Silva, J.R.O.: Towards historical R-Tree. In Proceesings of the 13th ACM Symposium on Applied Computing (ACM-SAC'98), 235-240, 1998.
[2] Nascimento, M.A., Silva, J.R.O, and Theodoridis, Y.: Evaluation for access Structures for discretely moving points. In Proceesings of the International Workshop on Spatio-Temporal Database Management (STDBM'99), Edinburgh, Scotland, 171-188, 1999.
[3] Tao, Y., and Papadias, D.: Mv3R-tree: a spatiotemporal access method for timestamp and intervalqueries. In Proceedings of the 27th International Conference on Very Large Databases, 431-440, 2001.
[4] Theodoridis, Y., Vazirgiannis, M., and Sellis, T.: Spatio-temporal Indexing for Large Multimedia Applications. In Proceedings of the 3rd IEEE Conference on Multimedia Computing and Systems, Hiroshima, Japan, 441-448, 1996.
[5] Pfoser D., Jensen C.S., and Theodoridis, Y.: Novel Approaches to the Indexing of Moving Object Trajectories. In Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 395-406, 2000.
[6] Elias frentzos.: Indexing object moving on fixed networks. Advances in Spatial and Temporal Databases: 8th International Symposium, SSTD 2003 Santorini Island, Greece, 289-305, July 24-27, 2003 Proceedings.
[7] Brinkhoff, T.: Generation Network-Based Moving Objects. In Proceedings of the 12th Int'l Conference on Scientific and Statistical Database Management, SSDBM'00, Berlin, Germany, 253-255, 2000.
[8] Y.Theodoridis, J. R. O. silva, and M.A Nascimento: On the Generation of Spatiotemporal Datasets. SSD, 147-164, 1999.
[9] A.Guttman: R-Tree: A Dynamic Index Structure for Spatial Searching. ACM SIGMOD, 47-57, 1984.
[10] N.Beckmann, H. Kriegel, R.Schmeider, and B. Seeger: The R*-Tree: An Efficient and Robust Access Method for Point and Rectangles. ACM SIGMOD, 322-331, 1990.

Publish/Subscribe Systems on Node and Link Error Prone Mobile Environments

Sangyoon Oh[1,2], Sangmi Lee Pallickara[2], Sunghoon Ko[1],
Jai-Hoon Kim[1,3], and Geoffrey Fox[1,2]

[1] Community Grids Lab., Indiana University, Bloomington, IN. U.S.A
{ohsangy, leesangm, suko, jaikim, gcf}@indiana.edu
[2] Department of Computer Science, Indiana University,
Bloomington, IN. U.S.A
[3] Graduate School of Information and Comminucations,
Ajou University, Suwon, S. Korea
jaikim@ajou.ac.kr

Abstract. Publish/subscribe model is appropriate in many push based data dissemination applications such as data dissemination services, information sharing, service discovery, etc. Recently, theses types systems are rapidly adopted in mobile and ubiquitous environment. However, mobile and ubiquitous environments are error prone due to wireless link disconnection, power exhaustion on mobile devices, etc. We analyze performance and effectiveness of publish/subscribe systems on the failure of client and server nodes and disconnection of communication links, which is common in mobile environments. We also perform experiments on our test bed to verify correctness and usefulness of our analysis.

1 Introduction

Publish/subscribe model [1] is appropriate in many applications such as data dissemination services [2], information sharing [3], service discovery [4], data dissemination services, information sharing, service discovery, etc. As these kinds of services are popular in mobile and ubiquitous environments, publish/subscriber model will be more widely used. Fig.1 depicts system configurations of publish/subscribe systems for error prone mobile environments. Publish/subscribe system consists of publisher (ES: Event Source), server (EBS: Event Brokering System), and subscriber (ED: Event Displayer). After publisher publishes data (events) asynchronously to a server, the server disseminates the data (events) to subscribers which registered its interest on the server. Main advantages of publish/subscribe systems include decoupling of publishers and subscribers in time, space, and synchronization [5-7]. Subscriber can access published data asynchronously anytime and anywhere at its own convenience. Also publisher, server, and subscriber can continue its operation even some parts of publish/subscribe system fails. These kinds of characteristics are main advantages of publish/subscriber model and make it useful in mobile ubiquitous environments [8].

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 576 – 584, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Many researches have been performed so far to propose architecture, add useful functions, and improve performance of publish/subscribe systems include Siena [9], Gryphon [10], JEDI [11], Rebeca [12], Elvin [13]. However, to our best knowledge, research of performance modeling for publish/subscriber system including node and communication link failures has not been announced yet. As mobile and ubiquitous service environments include many kinds of mobile devices and sensors which are connected via wireless networks, failure (or unavailability) rate of device and communication links are much higher than conventional wired environments. Thus, it is essential to consider how the failure and unavailability of mobile devices and wireless networks affect on performance and use of service. We analyze the influence of errors on mobile devices, servers, and wireless links, and compare to other interaction-based models such as client-server model and polling models. We can estimate performance in error prone environment and effectively adopt publish/subscribe systems by using our analysis. The results of our analysis show that publish/subscribe system is more durable than client/server models in error prone mobile and ubiquitous environments including mobile devices and wireless networks. We also perform some experiment on test bed to verify correctness of our analysis.

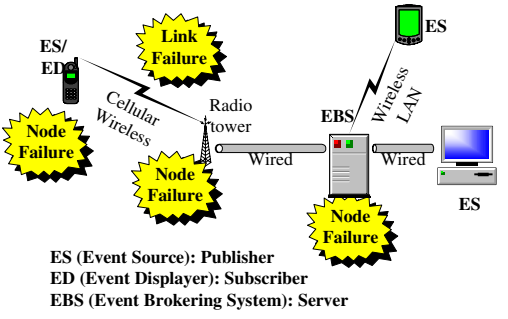


Fig. 1. Pub/Sub System Configurations

2 Cost Model

2.1 System Models

In this subsection, we propose cost analysis model for publish/subscribe systems. We assume following basic system parameters to analyze cost.

- α : publish rate
- β : subscriber's access rate of published events, or request rate of client in the client/server models
- c_{ps} (α) : publish/subscribe cost per event, c_{pub} (cost for publish events) + c_{sub} (cost for subscribe events)
- c_{rr} (β) : cost per request and reply in client-server model
- $c_{poll}(\alpha,T)$: cost of periodic publish or polling (We assume function of α and T (ex. $c_{pub}\alpha T$, c_{poll}), where T is length of period.)
- s(n) : effect of sharing among n subscribers, e.g., server can deliver events with low cost when it broadcasts event to many subscribers.
- t_{ps} : time delay for publish/subscribe, t_{pub} (time delay for publish) + t_{sub} (time delay for subscribe)
- t_{rr} : time delay for request and reply in request-reply (client-server) model

- $t_{poll}(\alpha, T)$: time delay for periodic publish
- $t_{proc}(\alpha, T)$: amount of processing time on server
- λ, λ_s, λ_c : failure rate of communication link, server, and client, respectively
- μ, μ_s, μ_c : recovery rate of communication link, server, and client, respectively.

2.2 Cost Analysis

We analyze cost of four different models, (1) publish/subscribe, (2) event(broker) message based request/reply, (3) request/reply, and (4) periodic polling models including failure (disconnection) and recovery (reconnection) of communication links as well as failure and recovery of node of server and client. Four different models can be categorized as shown in Table 1. Most implementations of publish/subscribe model and some request/reply models (we call it as "event message based request/reply") include *persistent file* and *durable database*, for fault-tolerance of server and client, respectively. *Persistent files* for events publish and subscribe are sharable among servers. When a server fails, the another server can take over the role and publish/server model can continue its transactions. *Durable database* saves events log and provides clients events history after clients recovers from failures. Without the durable database, clients could not receive events occurred during failure.

Table 1. Characteristics on Failures

Models	Types	Link failures	Server failures	Client failures
publish/subscribe	publish/ subscribe	Wait until link reconnection, t ransaction is preserved	Server is replicated an d transaction can be c ontinued with sharabl e *persistent file*	After recovery, client can access events occurred dur ing failure using events lo g on *durable database*
Event message ba sed request/reply	request/ reply	Wait until link reconnection, t ransaction is preserved	Server is replicated an d transaction can be c ontinued with sharabl e *persistent file*	After recovery, client can access events occurred dur ing failure using events lo g on *durable database*
request/reply (RPC)	request/ reply	Wait until link reconnection, t ransaction needs to restart	Transaction needs to r estart after recovery	Lost event or data during c lient failure
periodic polling	any	Wait until link reconnection	Depends on system	Depends on system

Cost metric in this analysis is time delay to transfer message and additional time required due to failure of communication links and nodes of server and client.

Publish/Subscribe Model

Failure of communication link: Besides cost to transfer message without disconnection (c_{pub} + n s(n)c_{sub}), additional cost (delay time) is required due to failure of communication links (disconnection). If communication link is disconnected during publish or subscribe, data transfer is delayed until the link is reconnected. There are two cases in disconnection: (1) communication link was connected but disconnected during message transfer for publish or subscribe is performed (probability is $\mu/(\lambda+\mu)$ (1- $\varepsilon^{-(t_{pub}+t_{sub})\lambda}$)), and (2) communication link was disconnected (probability is $\lambda/(\lambda+\mu)$). When disconnection occurs (probability is $\mu/(\lambda+\mu)(1-\varepsilon^{-(t_{pub}+t_{sub})\lambda})$+

$\lambda/(\lambda+\mu)$) and subscriber accesses events before reconnection (probability is $\beta/(\mu+\beta)$), average delay time due to link disconnection is $1/\mu$. Thus, cost of publish/subscribe model for each event publish and subscribe in the view point of a subscriber is:

$$t_{pub} + t_{sub} + \{\mu/(\lambda+\mu)\,(1-\varepsilon^{-(t_{pub}+t_{sub})\lambda}) + \lambda/(\lambda+\mu)\}\,\{\beta/(\mu+\beta)\}(1/\mu). \quad (1)$$

Failure of server: When a server (broker) fails, the another server (broker) can take over the role and publish/server model can continue its transactions without any interruption. We ignore the cost required for task transition from failed server to backup server. Thus, cost is:

$$t_{ps} = t_{pub} + t_{sub}. \quad (2)$$

Failure of client: After a client recovers from failure, client can obtain any event at anytime if durable database exists, which logs data (or events) to be used by client after recovery from failure. However, size of durable database is limited. If durable database can log maximum of n_{log} data (events), a client can be provided up to n_{log} recent events occurred before recovery. However, data is not available to the client after its recovery from a failure when the client requires more prior event than n_{log}-th recent event. The probability that i events occurred between failure and recovery is:

$$\left(\frac{\alpha}{\alpha+\mu_c}\right)^i \frac{\mu_c}{\alpha+\mu_c} \quad (3)$$

If $i \geq n_{\log} + 1$, i - n_{log} events are lost due to exceeding limitation of capacity for event log. Thus, an average number of lost events per client failure is:

$$\sum_{i=n_{\log}+1}^{\infty} \left(\frac{\alpha}{\alpha+\mu_c}\right)^i \frac{\mu_c}{\alpha+\mu_c}(i-n_{\log}) = \left(\frac{\alpha}{\alpha+\mu_c}\right)^{n_{\log}} \frac{\alpha}{\mu_c} \quad (4)$$

Event (Broker) Message Based Request/Reply Model

Failure of communication link: Besides cost (delay time) without disconnection (t_{rr}), additional cost (delay time) is required due to disconnection of communication link. If communication link is disconnected during request and reply, a transaction is delayed until the link is reconnected. There are two cases in disconnection: (1) communication link was connected but disconnected during a transaction (sending request message, processing the request on a server, and receiving reply) is performed (probability is $\mu/(\lambda+\mu)(1-\varepsilon^{-t_{rr}\lambda})$), and (2) communication link was disconnected (probability is $\lambda/(\lambda+\mu)$). When disconnection occurs (probability is $\mu/(\lambda+\mu)(1-\varepsilon^{-t_{rr}\lambda})+\lambda/(\lambda+\mu)$), average delay time is $1/\mu$. Thus, cost of request/reply model is :

$$t_{rr} + \{\mu/(\lambda+\mu)(1-\varepsilon^{-t_{rr}\lambda})+\lambda/(\lambda+\mu)\}(1/\mu) \quad (5)$$

Failure of server: When we assume that persistent file and durable database exist in event (broker) message based request/reply model, request/reply model can continue its transaction. The roles of persistent file and durable database are similar to those of publish/subscribe model. Persistent files in publish/subscribe model are sharable

among servers. When a server fails, the another server can take over the role and receives requests from clients and replies to clients. Thus request/reply model can con tinue its transactions without any interruption. We also ignore the cost required for task transition from failed server to backup server. Thus, cost is:

$$t_{rr} \tag{6}$$

Failure of client: Durable database saves information. Thus, a client can receive reply any information at anytime after it recovers from a failure up to n_{log}-th recent data. Thus, an average number of lost data item per client failure is the same as Equ. (3).

RPC Based Request/Reply Model (Non-persistent and Non-durable)

Failure of communication link: In addition to waiting time for link recovery, transaction of request and reply procedure will be restarted after the link recovers if persistent file does not exist. Thus, following additional costs (time delay) are required:

- useless computation: If a server fails a transaction of during request/reply process, a client needs to request the server again after recover. Thus, the previous request/reply process until the server failure is useless (lost). We call it useless computation. The amount useless computation is the maximum of $t_{rr} + t_{proc}$.
- recovery: When a server recovers from a failure, some amount of time is required for recovery procedure. In this analysis, recovery time is ignored like the other models (publish/subscriber and event (broker) message based request/reply models).

We analyze cost (time) delay of request/reply model on a link failure in three cases: (1) Link does not fail (probability is $1-\mu/(\lambda+\mu)(1-\mathcal{E}^{-(t_{rr}+t_{proc})\lambda})-\lambda/(\lambda+\mu)$): Required cost is t_{rr} , (2) Link was failed (probability is $\lambda/(\lambda+\mu)$): Cost is needed to wait until link recovers ($1/\mu$) and restart transaction (c_{rr}), and (3) Link fails during a transaction (probability is $\mu/(\lambda+\mu)(1-\mathcal{E}^{-(t_{rr}+t_{proc})\lambda})$): In addition to useless computation ($E(t_{rr}+t_{proc},\lambda)$), costs to wait until link recovers ($1/\mu$) and restart transaction (c_{rr})) are necessary. Thus, cost is:

$$\begin{aligned} c_{rr} &= [1-\{\mu/(\lambda+\mu)(1-\mathcal{E}^{-(t_{rr}+t_{proc})\lambda})+\lambda/(\lambda+\mu)\}]t_{rr} \\ &\quad + \mu/(\lambda+\mu)(1-\mathcal{E}^{-(t_{rr}+t_{proc})\lambda})\{E(t_{rr}+t_{proc},\lambda)+1/\mu+c_{rr}\}+\lambda/(\lambda+\mu)\{(1/\mu)+c(\beta)\} \\ \Rightarrow c_{rr} &= [\,[1-\{\mu/(\lambda+\mu)(1-\mathcal{E}^{-(t_{rr}+t_{proc})\lambda})+\lambda/(\lambda+\mu)\}]t_{rr} \\ &\quad + \mu/(\lambda+\mu)(1-\mathcal{E}^{-(t_{rr}+t_{proc})\lambda})\{E(t_{rr}+t_{proc},\lambda)+1/\mu\}+\lambda/(\lambda+\mu)(1/\mu)] \\ &\quad /[1-\{\mu/(\lambda+\mu)(1-\mathcal{E}^{-(t_{rr}+t_{proc})\lambda})+\lambda/(\lambda+\mu)] \end{aligned} \tag{7}$$

$E(x, \lambda)$ denotes the amount of time, during an interval of length x, before a failure occurs with rate of λ, given that failure rate is λ and the failure occurs during the interval. Then,

$$E(x, \lambda) = \int_0^x t \frac{\lambda \varepsilon^{-\lambda t}}{1-\varepsilon^{-\lambda t}} dt = \lambda^{-1} - \frac{x \varepsilon^{-\lambda x}}{1-\varepsilon^{-\lambda x}} \tag{8}$$

Failure of server: When persistent file does not exist, transaction of request/reply model is delayed on a server failure until the server recovers from the failure. Moreover, transaction of request and reply procedure will be restarted after the server recovers. Cost analysis is similar to that of failure of communication link (Equ. (5)). We analyze cost (time) delay of request/reply model on a server failure in three cases: (1) Server does not fail (probability is $1-\mu_s/(\lambda_s+\mu_s)(1-\varepsilon^{-(t_{rr}+t_{proc})\lambda_s})-\lambda_s/(\lambda_s+\mu_s)$): Required cost is t_{rr} , (2) Server was failed (probability is $\lambda_s/(\lambda_s+\mu_s)$): Cost is needed to wait until server recovers ($1/\mu_s$) and restart transaction (c_{rr})), and (3) Server fails during a transaction (probability is $\mu_s/(\lambda_s+\mu_s)(1-\varepsilon^{-(t_{rr}+t_{proc})\lambda_s})$): In addition to useless computation ($E(t_{rr}+t_{proc},\lambda_s)$), costs to wait until server recovers ($1/\mu_s$) and restart transaction (c_{rr})) are necessary. Thus, cost is:

$$\begin{aligned}
c_{rr} &= [1-\{\mu_s/(\lambda_s+\mu_s)(1-\varepsilon^{-(t_{rr}+t_{proc})\lambda_s})+\lambda_s/(\lambda_s+\mu_s)\}]t_{rr} \\
&\quad +\mu_s/(\lambda_s+\mu_s)(1-\varepsilon^{-(t_{rr}+t_{proc})\lambda_s})\{E(t_{rr}+t_{proc},\lambda)+1/\mu_s+c_{rr}\}+\lambda_s/(\lambda_s+\mu_s)\{(1/\mu_s)+c_{rr}\} \\
\Rightarrow c_{rr} &= [\,[1-\{\mu_s/(\lambda_s+\mu_s)(1-\varepsilon^{-(t_{rr}+t_{proc})\lambda_s})+\lambda_s/(\lambda_s+\mu_s)\}]t_{rr} \\
&\quad +\mu_s/(\lambda_s+\mu_s)(1-\varepsilon^{-(t_{rr}+t_{proc})\lambda_s})\{E(t_{rr}+t_{proc},\lambda)+1/\mu_s\}+\lambda_s/(\lambda_s+\mu_s)(1/\mu_s)] \\
&\quad /[1-\{\mu_s/(\lambda_s+\mu_s)(1-\varepsilon^{-(t_{rr}+t_{proc})\lambda_s})+\lambda_s/(\lambda_s+\mu_s)]
\end{aligned} \tag{9}$$

Failure of client: As we assume that a server of RPC based request/reply model contains a current data (or event) only without logging data during a failure of client, data (event) occurred during the failure is lost.

Periodic (Polling) Model

In the periodic model, the period of publish (or polling) is delayed until communication link is reconnected once communication link is disconnected. Cost of periodic model due to disconnection is function of delayed probability and time. When link was disconnected or is disconnected during message delivery (probability of $\mu/(\lambda+\mu)\,\varepsilon^{-t_{poll}(\alpha,T)\lambda}+\lambda/(\lambda+\mu)$), period is delayed by $1/\mu$ on average. Thus time delay is:

$$t_{poll}(\alpha,T) + \{\mu/(\lambda+\mu)\,\varepsilon^{-t_{poll}(\alpha,T)\lambda} + \lambda/(\lambda+\mu)\}(1/\mu) \tag{10}$$

If we consider conceptual cost:

$$\begin{aligned}
&c_{poll}(\alpha,T) + [1-\{\mu/(\lambda+\mu)\,\varepsilon^{-t_{poll}(\alpha,T)\lambda}+\lambda/(\lambda+\mu)\}]c_{delay}(\alpha,T) \\
&+\{\mu/(\lambda+\mu)\,\varepsilon^{-t_{poll}(\alpha,T)\lambda}+\lambda/(\lambda+\mu)\}c_{delay}(\alpha,T+1/\mu)
\end{aligned} \tag{11}$$

We also assume that persistent file and durable database exist in persistent and durable periodic model. Costs are basically same as Equ's (5) and (6) in the case of server failure of communication links and server, respectively, and Equ (3) in client failure.

3 Performance Comparisons

In this section, we describe performance comparisons by parametric analysis. System parameters are setup as in the Table 2. Curves in Fig.2 show time delay when communication links are transiently disconnected and reconnected. As publisher and subscriber are decoupled in time and space, publish/subscribe system is more tolerable than request/reply system from failure. Publish/subscribe system requires less communication delay in compare to request/reply system. We also measure communication delay and additional service time delay per transaction due to sever failure. As shown in Fig.3, publish/subscribe system (curve "pub/sub") and event message based request/reply system (curve "req/reply") always require only communication delay since backup server takes over failed server without transition time. However, RPC based request/reply system (curve "RPC") requires additional service time delay and lost

Table 2. Parameters

Param.	Values
α, β	0.5
c_{ps}, c_{rr}	2
c_{pub} , c_{sub}	1
$c_{poll}(\alpha, T)$	1 or α T
$c_{delay}(\alpha,T)$	0, T, or α T
s(n)	1/n - 1
λ , λ_s, λ_c	0.0001 – 0.5
μ, μ_s	0.1
μ_c	0.05 - 0.1
t_{ps} , t_{rr}	1
t_{proc}	1 or 5
$t_{poll}(\alpha, T)$	1, T, or α T

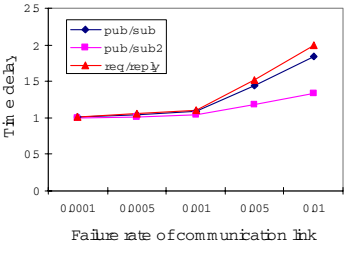


Fig. 2. Time delay per transaction links (α=0.5, s(n)=1, t_{ps}=1, t_{rr}=1, μ=0.1, and β=0.5 for pub/sub and req/reply and β=0.2 for pub/sub2)

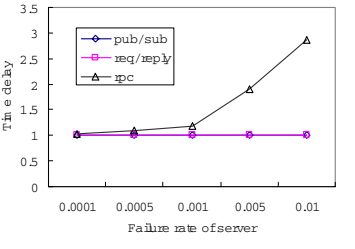


Fig. 3. Time delay per transaction (α=0.5, t_{ps}=1, t_{rr}=1, β=0.5, and μ_s=0.1)

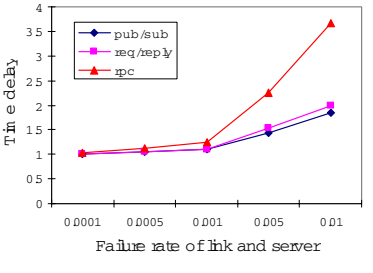


Fig. 4. Time delays per transaction (α=0.5, t_{ps}=1, t_{rr}=1, t_{poll}=1, β=0.5, μ=0.1, μ_s=0.1)

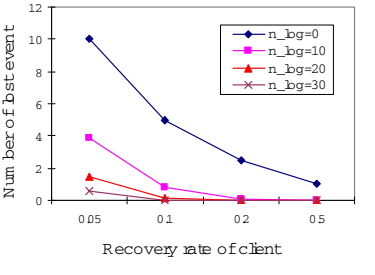


Fig. 5. Number of lost events per client failure by varying recovery rate of client

Table 3. Experimental results: Delay time of sending message

	Wireless	Wired	Total (msec.)
RPC	1290.7 (client – gateway)	39.9 (gateway – server)	1330.6 (t_{rr})
Pub/Sub	1448.4 (ED – EBS)	89.7 (EBS – ES)	1538.1 (t_{ps})

computation due to server failure. We also measure performance considering both failures of communication link and server. Fig.4 depicted results. As we expected, performance of "RPC" is the worst. Fig.5 shows effectiveness of durable database which logs events for the failure of client. Without logging, a client loses events occurred during its failure. As shown in Fig.5, number of lost events can be reduced as maximum number of log increases. We can consider "nlog=0" curve as publish/subscribe system without durable database.

Table 3 shows our experimental results (time delay) of sending short message (10 bytes) in a practical environment. The experiment environment consists of NaradaBrokering [14] system where a HHMS (HandHeld Message Service) Proxy [15] plugged in, mobile clients, and conventional PC applications. We have an experiment with a J2ME echo client application and a simpleserver that sends 10 byte message. The round trip time is measured 2000 times and the median value is taken. We use Treo 600 with Sprint PCSVision service as a client device. In our analysis, we set parameters $t_{ps} = t_{rr} = 1$, which is comparable to our experimental results (t_{rr} = 1.33 sec. and t_{ps} =1.53 sec.). We need to adjust various parameters for each system because these parameters depend on systems and application environments.

4 Conclusion

Publish/subscribe model has many advantages in push based mobile applications. However, mobile and ubiquitous environments are error prone due to wireless link disconnection, power exhaustion on mobile devices, etc. We analyze performance and effectiveness of publish/subscribe systems on the failure of client and server nodes and disconnection of communication links. The results of our analysis show that publish/subscribe system is more durable than client/server models in error prone mobile and ubiquitous environments including mobile devices and wireless networks.

References

1. P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec, "The Many Faces of Publish/Subscribe," ACM Computing Surveys, vol. 35, no. 2, Jun. 2003, pp. 114-131.
2. G. Muhl, A. Ulbrich, K. Herrmann, and T. Weis, "Disseminating Information to Mobile Clients Using Publish-Subscribe," Proc. of the IEEE Internet Computing, Vol.8, No. 3, 2004
3. Sagar Chaki, Pascal Fenkam, Harald Gall, Somesh Jha, Engin Kirda, and Helmut Veith, "Integrating Publish/Subscribe into a Mobile Teamwork Support Platform," Proc. of the 15th International Conference on Software Engineering and Knowledge Engineering 2003

4. Zhexuan Song, Yannis Labrou and Ryusuke Masuoka, "Dynamic Service Discovery and Management in Task Computing," Proc. of the 1st International Conference on Mobile and Ubiquitous Systems: Networking and Services, August 22-25, 2004
5. L. Fiege, F. Gartner, O. Kasten, and A. Zeidler, "Supporting Mobility in Content-Based Publish/Subscribe Middleware," Middleware 2003, LNCS 2672, pp. 103 – 122, 2003.
6. M. Caporuscio, A. Carzaniga, and A. Wolf, "Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications," IEEE Transactions on Software Engineering, vol. 29, no. 12, pp. 1059 – 1071, Dec. 2003.
7. U. Farooq, E. Parsons, and S. Majumdar, "Performance of Publish/Subscrive Middleware in Mobile wireless Networks," Proc. of WOSP'04, pp. 278-289, Jan. 2004.
8. S. Lee, S. Ko, G. Fox, K. Kim, and S. Oh, "A Web Service Approach to Universal Accessibility in Collaboration Services", Proc. of ICWS, Las Vegas June 2003.
9. A. Carzaniga, D.Rosenblum, and A. Wolf, "Design and evaluation of a wide-area event notification service," ACM Transactions on Computer Systems, 2001
10. M. Aguilera, R. Strom, D. Sturman, M.Astley, and T. Chandra, "Matching events in a content- based subscription system," Proc. of ACM Symp. on Principles of Distributed Computing, 1999
11. G. Cugola, E. Di Nitto, and A. Fuggeta, "The JEDI Event-based infrastructure and its Application to the Development of the OPSS WFMS," IEEE Trans. of Software Engineering, 2001
12. L. Fiegen, G. Muhl, and F. Gartner, "A Modular Approach to Building Event-Based Systems," Proc. of ACM Symposium on Applied Computing, 2002
13. B. Segall, D. Arnold, J. Boot, M. Henderson and T. Phelps, "Content Based Routing with Elvin4," Proc. of AUUG2K, 2000
14. Shrideep Pallickara and Geoffrey Fox, "NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids," Proc. of ACM/IFIP/USENIX International Middleware Conference Middleware, pp 41-61, 2003.
15. Sangyoon Oh, Geoffrey C. Fox, Sunghoon Ko, "GMSME: An Architecture for Heterogeneous Collaboration with Mobile Devices", Proc. of MWCN 2003, Singapore, Oct., 2003.

A Power Efficient Routing Protocol in Wireless Sensor Networks

Hyunsook Kim, Jungpil Ryu, and Kijun Han*

Department of Computer Engineering,
Kyungpook National University, Korea
{hskim, goldmunt}@netopia.knu.ac.kr, kjhan@knu.ac.kr

Abstract. Saving energy consumption for a long network lifetime is one of the most important issues in design of wireless sensor networks because they usually consist of a large number of nodes with limited battery power. In this paper, we propose a balanced tree construction scheme called BATR (Balanced Aggregation Tree Routing), which uses near optimal minimal spanning tree for balancing the power consumptions over all nodes. Our main idea is that if energy consumed for transmission and reception can be nicely balanced over all nodes, an optimal data aggregation can be achieved in terms of network lifetime. Simulation results assure that the BATR can lengthen system lifetime as compared with the conventional data aggregation protocols.

1 Introduction

Recent advancements in wireless communication and highly progressed hardware technology have enabled the development of a wireless sensor with a low-cost processor, low-power, and light-weight. Each sensor collects useful information from the region in a variety of scenarios including military surveillance and landmine detection in harsh physical environments and monitoring [1]. Typically, sensor nodes collect audio, seismic, and other types of data and collaborate to perform a high-level task in a sensor web. Since wireless communications consume a significant amount of battery power, sensor nodes should be energy efficient in transmitting data. Energy efficient communication in wireless networks is attracting increasing attention in the literature.

Since sensor nodes might generate significant redundant data, data aggregation is usually used for wireless routing in sensor networks [2][3]. In each round of this data-gathering application, all data from all nodes need to be collected and transmitted to the sink, where the end-user can access the data [9]. There are also different protocols proposed in literatures ([7][8]) so as to maximize the lifetime of the system under different circumstances. The simplest approach to routing in wireless sensor networks is direct transmission, in which each node transmits its own data directly to the sink.

In direct transmission, if the location of sink is far from sensor field, each sensor node dissipates a large amount of energy.

* Correspondent author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 585 – 592, 2005.
© Springer-Verlag Berlin Heidelberg 2005

LEACH(Low-Energy Adaptive Clustering Hierarchy)[9] is proposed in order to efficiently do data gathering and aggregation. LEACH is a clustering-based protocol that utilizes randomized rotation of the cluster-heads to evenly distribute the energy load among the sensor nodes in the network. It assumes that the sink is fixed and located far from the sensors, and all nodes in the network are homogeneous and energy-constrained. But, LEACH is impractical, since it is directly transmission method. Thus, It is desirable to multi-hop transmit in wireless sensor network In [10], the authors propose a new minimum spanning tree-based protocol, called PEDAP (Power Efficient Data Gathering and Aggregation Protocol) and its power-aware version (PEDAP-PA). In PEDAP, the locations of all nodes are known by sink a priori, and path information of network is computed by sink. PEDAP is based on minimum spanning tree routing scheme as well. These previous research and concurrent works on protocol and algorithm designed in sensor networks are summarized in [1][4][5].

In this paper, we propose a balanced-tree based routing protocol called BATR (Balanced Aggregation Tree Routing) to achieve an optimal data aggregating in terms of the network lifetime by balancing power consumption per node. In our scheme, each intermediate node keeps the number of children nodes and it transmits the data to the nearest neighbor node. Our scheme may minimize the energy consumption in each node, thus prolong the lifetime of the system regardless of location of the sink is outside or inside the sensor field.

The rest of the paper is organized as follows. The BATR protocol is described in detail in Section 2. Next, in Section 3, we present our simulation results. Finally, we conclude the paper and present future research directions in Section 4.

2 Balanced Aggregation Tree Routing Protocol

2.1 System Model

Data aggregation, which is the combination of data from different multiple sources, is needed for saving energy by reducing the number of messages in the network. In general, data gathering requires some energy at intermediate nodes. To investigate power consumption in wireless sensor network, we use a radio model as described in [12], which is the first free space channel model where the power loss d^2 is used. First, we formally model the network as a graph G = $\{V,E\}$ where V is the set of nodes $\{0,\ldots..,n\}$ in the network and E is set of edges between nodes. We assume that combining n packets of size k results in one packet of size k instead of size nk [10]. We define the edges by $\forall x \, and \forall y \in V, \exists (x, y) \in E \Leftrightarrow dist(x, y) \le d$, where d is the transmission range of sensor nodes in the network and dist (x , y) is the Euclidean distance between nodes x and y which are said to be neighbors of each other. The set of neighbors of a node X is denoted as neighbors (X). For a tree T constructed for aggregation tree from graph G, we call a set $X \subseteq V$, a group iff $X = \{Y\}_{Y \, Z}$ where Y is the parent of nodes of Z in the tree T [11].

To transmit a k-bit packet over a distance d, the radio expends

$$E_{Tx}(k, d) = kE_{elec} + kd^2 \varepsilon_{fs} \tag{1}$$

to receive this message, the radio expends

$$E_{Rx}(k) = kE_{elec} \tag{2}$$

In these radio parameters, E_{elec} means a radio dissipates to run the transmitter or reciever circuitary and ε_{fs} for the transmitter amplifier separately.

As seen above, since the energy cost consists of two components, one for transmitting or receiving a packet in the radio electronics and the other depending on the distance in transmissions, the routing protocols must minimize the number of data receptions and while minimizing the transmit distance for a specific node. In order to maximize the network lifetime, the total energy should be minimized in a round of communication, while balancing the energy consumption among the nodes. An ideal aggregation tree can be thought as a balanced tree with a minimum number of edges, equivalent children nodes in order to maximize the network lifetime. For this, our protocol is designed to find an optimal path based on a balanced tree that has almost the same number of children nodes below itself. Here, the child node means the node that could exist in any neighborhood.

To describe our protocol, we assume that the system consists of *n* sensors distributed randomly in the environment that is being monitored. We assume that sink knows the locations of all nodes priori either by manually entering coordinates or by using the GPS. Each node periodically senses its nearby environment and would like to send its data to a sink. The sensor nodes, including the sink in the network have equal transmission range (say *d*). The nodes aggregate or fuse the data they receive from the others with their own data, and produce only one packet regardless of how many packets they receive. In addition, it is assumed that all nodes generate equal amount of data, and there is no mobility in the network[10].

2.2 Detailed Procedure

In our protocol, the routing information is computed using a MST(Minimum Spanning Tree) algorithm. Given a topology graph *G(V, E)*, it begins with the root node and starts iteratively establishing parent-child relationship with other nodes. Sink functions as a root node in MST.

(1) Initially, we start the path construction from sink as a root.
(2) After that, we select the nearest nodes among neighbors as many as the number of decided children nodes from a vertex in the tree to a vertex not in the tree, and add that edge to the tree. In our case, data will be send through that edges just included in the tree from each node. In our protocol, we balance the number of children nodes depending on the density of network to achieve a long lifetime. Here, the number of children nodes, denoted by μ, is given as an operational parameter whose value can be calculated by

$$\mu(R) = \alpha \frac{N\pi R^2}{A} \tag{3}$$

where *N* is the number of scattered sensor nodes in region *A*, and *R* can be either the range of a particular sensor or the radio transmission range (idealized with circular propagation) [13]. Basically, $\mu(R)$ gives the number of nodes within

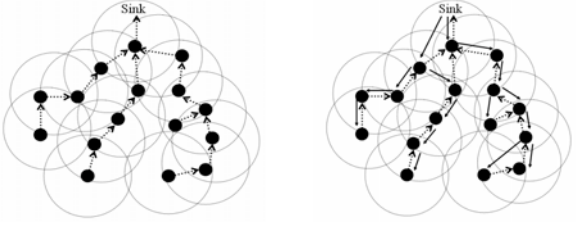
the transmission radius of each node in region A. α is a variable which depends on the position of sink. If the number of children nodes is a, energy consumption of intermediate node for data aggregation is

$$E_{\text{intermediate}}(k,a,d) = kE_{elec} + kd^2\varepsilon_{fs} + a(kE_{elec} + kE_{DA}) \quad (4)$$

So, the total energy expended by the network is given by the sum of energies used by each node ($E_{Total} = \sum_{i \in N} E_i$).

(3) When neighbor node is found over the limit of search, the node is labeled as a leaf node. We set the search limit at 15 for prevention of delay.
(4) We repeat this procedure until all available nodes are added to the tree.
(5) After some certain number of rounds, the sink recomputes the routing information considering the condition of nodes and resetting the number of children nodes. After each computation, the sink sends each node the required routing information for that node such as the node's parent in the tree in order to reach to the sink and the number of child nodes, etc.

Fig. 1 shows the process of tree construction and the result of routing paths for a sample network when children nodes are under two. A dotted line indicates the path which is node's aggregation information including death node. And solid line shows the resulting routing paths for a sample network.



(a) Aggregation of node's information (b) Result of routing paths

Fig. 1. An example for construction of a balanced tree

3 Simulation Results

We use the simulation programs written in C++ program language to evaluate the performance of our scheme. Our scheme is compared with direct transmission, LEACH, and non-balanced routing scheme which uses Prim's spanning tree algorithm. The network dimensions for our experiments is 100×100 and 100 nodes are randomly generated within the region. We run the simulations with different position of sink: to the center of the field and to the place away from the region.

Table 1. gives the parameters for the energy dissipation of the radio in order to run the transmitter or receiver circuitry.

Table 1. Parameters for simulation

Description	Parameter	Value
Radio electronics	E_{elec}	50 nJ/bit
Transmit amplifier	ε_{fs}	100 pJ/bit/m^2
Aggregation cost	E_{DA}	5 nJ/bit/signal
Data size	D_{size}	1000 bits

Fig. 2 shows the number of rounding of death nodes for networks where the sink is located in the center of the field. In general, the direct transmission can offer a satisfactory performance when the transmission distance between the node and the sink is not so far. However, BATR shows a similar performance to the direct transmission after the number of dead nodes reaches 50%. We can predict this result from Eq. (4) since there is no child node when more than 50% of nodes are dead. Taken as a whole, BATR improves the lifetime about 1.5 times compared with non-balanced tree protocol.

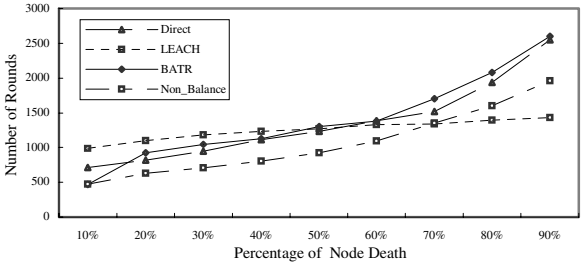


Fig. 2. Lifetime for a 100×100 network where the sink is at the center of the field

Fig. 3 indicates that BATR is the best power saving system when sink is far away from the field. As seen in Fig. 4, the lifetime obtained with direct transmission is very short since if the sink is far away from nodes, the nodes will consume more energy for data transmission and so die quickly. In case of non-balanced tree, distant nodes should send data directly after a certain time since nodes around the sink die first. On the other hand, dead nodes will be distributed uniformly over the network in BATR since the distances among nodes are not so far.

Fig. 4(a) shows that the lifetime as the number of children nodes is varied. It is worth to note that our scheme also performs well when the sink is outside the field. As shown from Eq. (1), energy consumption of each node is proportional to the square of distance between nodes. When the number of children nodes is incremented, the depth of tree is decreased but the distance between nodes becomes large. So, each node will consume more energy at this time. In our experiments, we can see that the longest lifetime can be achieved when there are two children nodes in case that sink is located far away from the field. Further, as we increase the number of children nodes, the network lifetime becomes shortened.

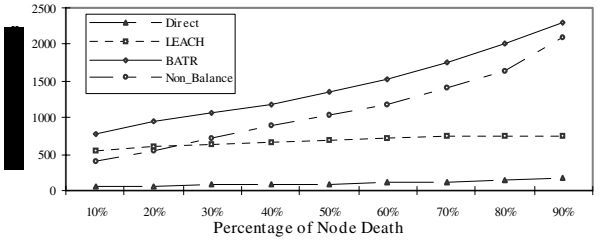
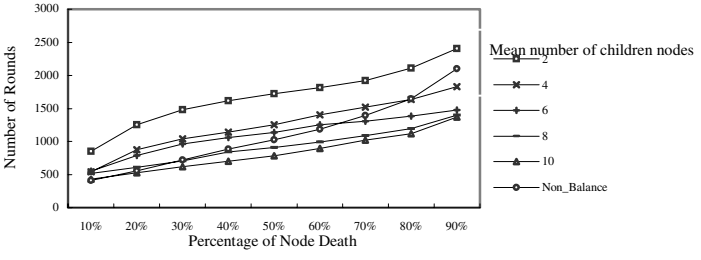
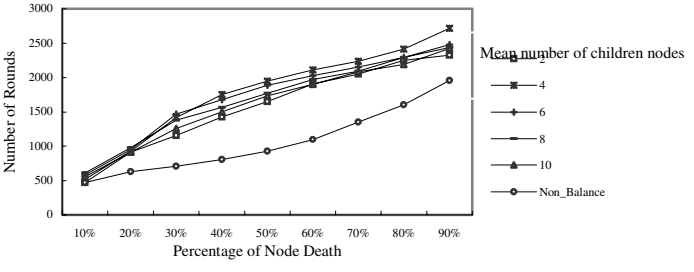


Fig. 3. Lifetime for a 100×100 network where the sink is outside of the network



(a) When the sink is located at the center of the field

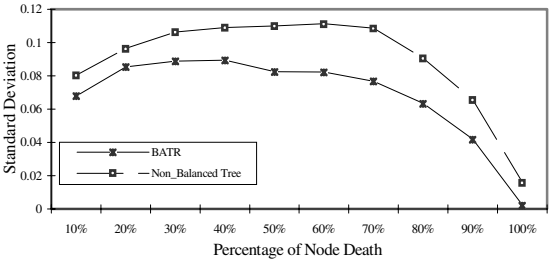


(b) When the sink is located outside of the field

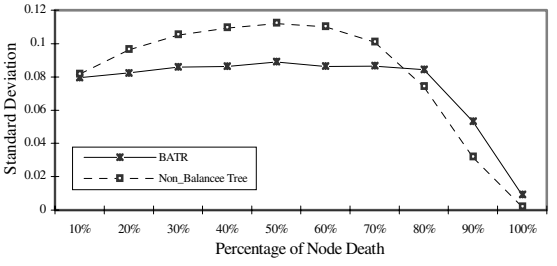
Fig. 4. Lifetime versus the number of children nodes

Fig. 4(b) represents a lifetime versus the number of children nodes when sink is located at the center of the field. We can observe almost the same lifetime regardless of the average number of children nodes. This is because the number of children nodes does not affect the energy consumption so much while the distance between nodes is the most dominant factor to the energy consumption.

To evaluate distribution characteristics of energy consumption in the network, we measure standard deviation of loads over all sensor nodes after each round. A large variance in load signifies that load is not uniformly distributed among the nodes. Fig. 5 demonstrates that our approach outperforms the non-balanced tree protocols.



(a) When the sink is located outside of the field



(b) When the sink is located in the center of the field

Fig. 5. Standard deviation of loads as the number of dead sensors is increased

As previously described, the energy cost consists of two components, one is the number of children nodes and the other is transmission distance. Therefore, the routing protocols must minimize the number of data receptions and reduce the distance of transmission between specific nodes. In BATR, a node with large transmission distance becomes a leaf node on a tree. So, it can save energy because it does not need aggregation and receiving. In case of intermediate nodes, the deviation of energy consumption can be reduced by balancing the number of children nodes.

4 Conclusion and Future Works

Distributing the load evenly to the nodes has a great impact on system lifetime. In this paper, we propose a power efficient routing protocol based on a balanced tree. Our main idea is minimizing the total energy of the system by balancing of energy consumption among the nodes. We show through simulations that our scheme performs near optimal regardless of the position of the sink and outperforms the conventional approaches such as LEACH and non-balanced tree protocols.

Our future work involves several refinements and extensions of this work, as well as establishment of analytical models to get more comprehensive underlying performance limit.

Acknowledgement

This research is supported by Program for the Training of Graduate Students for Regional Innovation.

References

1. I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine,* Vol. 40, No. 8, pp. 102-116, August 2002.
2. J.Heidemann, F.Silva, C.Intanagonwiwat, R. Govindan, D.Estrin, and D. Ganesn, "Building Efficient Wireless Sensor Networks with Low-Level Naming", *18th ACM Symposium on Operating Systems Principles*, October 21-24,2001
3. C. Intanagonwiwat, R.Govindan and D.Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *ACM/IEEE International Conference on Mobile Computing and Networks(MobiCom2000)*, August 2000, Boston, Massachusetts
4. J.-H. Chang and L.Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *IEEE INFOCOM 2000*, March 2000, pp 22-31.
5. J. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," in *Proceedings of Advanced Telecommunications and Information Distribution Research Program, College Park*, MD, 2000
6. M. Bhardwaj, A. Chandrakasan, and T. Garnett, "Upper bounds on the lifetime of sensor networks," in *IEEE International Conference on Communications*, 2001, pp. 785-790.
7. W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *MOBICOM*, 1999, pp. 174–185.
8. K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum lifetime data gathering and aggregation in wireless sensor networks," in *Proceedings of the 2002 IEEE International Conference on Networking (ICN'02)*, August 2002, pp. 685–696.
9. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy efficient communication protocol for wireless microsensor networks," in *33rd Annual Hawaii International Conference on System Sciences,*, 2000, pp. 3005 – 3014.
10. Hüseyin Özgür Tan and Îbrahim Körpeogle , "Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks," in *SIGMOD Record,* Vol, 32, No. 4. Dec 2003,
11. V. Annamalai, SKS and Gupta L. Schwiebert, "On Tree-based convergecasting in wireless sensor networks," *IEEE Wireless Communications and Networking Conference(WCNC)* ,pp 1942-1947, March 2003
12. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications,* Vol. 1, No. 4, Oct 2002
13. Bulusu et al., "Scalable Coordination for Wireless Sensor Networks : Self-Configuring Localization Systems," *ISCTA 2001,* Ambleside, U.K., July 2001

Applying Mobile Agent to Intrusion Response for Ad Hoc Networks

Ping Yi, Yiping Zhong, and Shiyong Zhang

Department of Computing and Information Technology Fudan University, 200433, China

Abstract. Mobile ad hoc networking offers convenient infrastructureless communication over the shared wireless channel. However, the nature of ad hoc networks makes them vulnerable to security attacks. Existing security technologies such as intrusion prevention and intrusion detection are passive in response to intrusions in that their countermeasures are only to protect the networks and there is no automated, network-wide counteraction against detected intrusions. While they all play an important role in counteracting intrusion, they do not, however, effectively address the root cause of the problem – intruders. In this paper, we propose the architecture of automated intrusion response. When an intruder is found in our architecture, the block agents will get to the neighbor nodes of the intruder and formed the mobile firewall to isolate the intruder.

1 Introduction

Mobile Ad hoc Network is an autonomous system of mobile nodes connected by wireless links. Each node operates not only as an end-system, but also as a router to forward packets. The nodes are free to move about and organize themselves into a network. Mobile ad hoc networks does not require any fixed infrastructure such as base stations, therefore, it is an attractive networking option for connecting mobile devices quickly and spontaneously, such as military applications, emergent operations, personal electronic device networking, and civilian applications like an ad-hoc meeting or an ad-hoc classroom.

With more and more application, security for mobile ad hoc networks becomes increasingly important. Existing security approaches for networks can be categorized as intrusion prevention, intrusion detection and intrusion response. Intrusion prevention utilizes authentication, encryption and firewall etc. to protect system from being attacked and compromised. However, some way may not be applied to mobile ad hoc networks, such as firewall. Because of dynamic topology, mobile ad hoc networks do not have a clear line of defence. When nodes roam in a hostile environment with relatively poor physical protection, they have probability of being compromised. The compromised nodes may launch attacks within the networks. Encryption and authentication can not defend against compromised nodes, which carry the private keys. Therefore, these approaches of intrusion prevention which can play an important role in traditional networks may take limited affection in mobile ad hoc networks.

Intrusion Detection Systems (IDS) attempts to detect intrusion by analyzing observed system or network activities. The IDS may be classified based on the detection technique, such as anomaly detection, misuse detection and specification-based detection. IDS will raise alarms when it has detected an attack.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 593 – 600, 2005.
© Springer-Verlag Berlin Heidelberg 2005

While intrusion prevention and detection all play an important role in solving the problem of today's network-based intrusion, they are all passive and not adequate to solve the intrusion problem. One fundamental problem with existing intrusion prevention, detection and tolerance is that they do not effectively eliminate or deter network-based intrusions. The best they can do is to avoid being victims of network-based intrusions temporarily. Because they do not address the root cause of intrusions – intruders, those intruders can always explore new system vulnerabilities, find new accomplices – potentially insiders, and launch new attacks from different places. Because intrusion prevention and detection do not effectively address the problem of compromised system recovery, intruders can use those compromised system as new base for further intrusion. What we need is an effective way to hold network-based intruders accountable for their intrusions, and an automated way to prevent compromised hosts from doing further harm.

So far most computer security research regarding intrusions has focused on prevention and detection. Intrusion response has been an afterthought and is generally limited to logging, notification. There are more vulnerability in mobile ad hoc networks for its wireless link and mobile node. If no response as soon as early, intruder may destroy entire networks. In addition, for self-organizing and lack of centralized administration, manual response is difficult to bring into effect, especial in multiple administrative domains.

In this paper, we propose Mobile Firewall to address the problem of intrusion. Our goal is to effectively and rapidly eliminate intrusion and reduce the damage as soon as possible by means of automated intrusion response. We make three contributions to the area of intrusion response for mobile ad hoc networks.

- Mobile Firewall: In a traditional network, a firewall is installed at the ingress point of the network to filter out unallowable traffic originating from outside the boundary of the protected network. In the paper, the Mobile Firewall is used to surround and isolate the intruder. The Mobile Firewall can move with the intruder node and isolate the intruder for ever.
- Shrapnel way to dispose: An artillery shell containing metal balls fused to explode in the air above enemy troops. By analogy, agent moves to the node around the intruder and reproduces a lot of agents to surround and isolate the intruder in the end.
- Local repair: when the intruder is found, all paths via the intruder will find a new route to bypass the intruder by the process of local repair.

The rest of the paper is organized as follows. Section 2 gives the related work. Section 3 addresses the approach of automated intrusion response. And section 4 concludes the paper.

2 Related Work

The Boeing Phantom Works carries out some researches in autonomic intrusion response. D. Schnackenberg et al present the Cooperative Intrusion Traceback and Response Architecture (CITRA) who integrated networks-based intrusion detection systems, firewalls, and routers to trace attacks back to their true source and block the

attacks close to that source [1]. D. Schnackenberg et al provide an overview of the Intruder Detection and Isolation Protocol (IDIP) [2]. Dan Sterne et al present the results of testbed experiments using CITRA and IDIP to defend distributed denial of service attacks [3]. The CITRA and IDIP support automatically tracing attempted intrusions across network boundaries and blocking or otherwise responding to them near their sources. This technology will enable networks of networks to cooperatively detect system attacks, exchange information about attack behaviour, and respond to attacks by dynamically reconfiguring routers, firewalls, and hosts to heighten their defensive posture and provide additional protection against subsequent attacks. The heart of this technology is a new protocol, the Intruder Detection and Isolation Protocol (IDIP), which enables cooperation among such components. Working together in internet worked systems, IDIP-enabled components can locate, isolate, and block an intruder close to the point of attack and provide diagnostic information so that network administrators can further investigate the intrusion. The above technology can take effect in traditional wireline networks environment, but they are not applied to mobile ad hoc networks for the wireless link and dynamic topology.

X. Wang et al propose Tracing Based Active Intrusion Response (TBAIR) [4]. Based on Sleep Watermark Tracing (SWT), TBAIR can trace the detected intrusion and push the intrusion countermeasures close to the source of the intrusion. N. Foukia et al present an approach inspired by natural systems for intrusion detection and response [5]. Its approach uses the immune system metaphor for intrusion detection agents and social insect's behavior metaphor for intrusion response. CA Carver et al propose a methodology for adaptive intrusion response using intelligent agents [6]. Jansen W et al propose a number of innovative ways to apply agent to address the shortcomings of current intrusion detection system [7]. C. A. Carver, and U. W. Pooch provide a categorization of possible offensive and defensive responses.

The papers of mobile ad hoc networks security can be classified in three categories: key management, secure network routing, and intrusion detection. Capkun, Buttyan and Hubaux propose a fully self-organized public key management system that can be used to support security of ad hoc network routing protocols [8]. Zhou and Hass first proposed to use threshold cryptography to securely distribute the Certificate Authority private key over multiple nodes to form a collective CA service[9]. Routing security has been most noted by its absence early in the discussion and research on ad hoc routing protocols. Since then several ad hoc routing protocols that include some security services have been proposed: SRP[10], Ariadne[11], ARAN[12], SEAD[13]. SRP[10] assumes the existence of shared secrets between all pairs of communicating nodes and leverages this for MAC authentication, such that fake route requests are not accepted at the destination and routes set in route replies cannot be modified. In Ariadne[11], end-to-end authentications are got by one-way hash chain and MAC authentication. ARAN[12] relies on public key certificates to retain hop-by-hop authentications. SEAD[13] use elements from a one-way hash chain to provide authentication for both the sequence number and the metric in each entry. Yongguang Zhang developed an Intrusion Detection architecture and evaluated a key mechanism in this architecture, anomaly detection for mobile ad-hoc networks [14]. These security systems only passively detect and prevent invader, they can not effectively eliminate or deter intrusions.

3 The Approach of Automated Intrusion Response

3.1 System Architecture

An agent is a small intelligent active object which is able to carry out activities continuously and autonomously in a particular environment. Agent is autonomous, lightweight, adaptive, and mobile. Multi-agent can communicate and cooperate with each other. These qualities make agent a choice for security architecture in bandwidth and computation-sensitive mobile ad hoc networks.

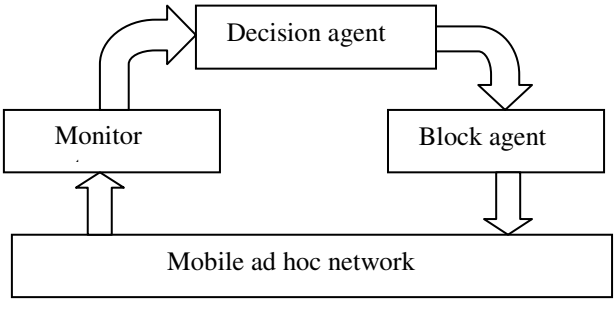


Fig. 1. System architecture

Figure 1 shows the three essential components of the architecture: monitor agent, decision agent and block agent. Monitor agent resides on each node and monitors the neighbour nodes by collecting all packets with its communication range. Monitor agent code behaviour information of its neighbour nodes and send them to decision agent. Decision agents collect the information from the monitor agent, and make a judgment by security policies. If the decision agent can judge that some node is invader, it will produce block agents. These block agents move to the neighbour nodes of the intruder and construct Mobile Firewall to surround the invader and isolate the intruder in the end. The whole response process is automated without manual intervention.

3.2 Mobile Firewall

In a traditional network environment, a firewall is installed at the ingress point of the network to filter out unallowable traffic originating from outside the boundary of the protected network. In mobile ad hoc wireless environment where the nodes could be mobile and topology of networks is dynamic, there is therefore no notion of a well defined ingress point for the network. In addition, any node within the network could become an intruder for compromised and so attack traffic could originate from within the network itself. Hence, the traditional firewall concept does not work in an ad hoc network. Furthermore, traditional firewalls are not designed to protect against spoofed packet flooding attacks where the attack traffic masquerades as legitimate packets passed through by the firewall access control rules.

The Mobile Firewall is used to protect mobile ad hoc networks against attacks launched from an intruder node within the networks. Mobile ad hoc networks are multi-hop wireless networks, and the node sends and receives packets through its neighbor nodes. If all neighbor nodes around the node refuse to forward its packets, the node can not communicate with the other nodes in mobile ad hoc networks. The node has been isolated from the network in practice even if it is still in the networks in location. The Mobile Firewall takes effect using the above principle. When decision agent finds an intruder, decision agent sends block agents to the neighbor nodes of intruder. These block agents reside in the nodes around the intruder and surround the intruder. In the mean time, they cut off the links between the nodes and the intruder, and they do not receive and forward any packet from the intruder. In the end the intruder is isolated from network. The traditional firewall is to protect the networks in firewall against outside attack, and the Mobile Firewall is to surround the intruder and protects the other nodes in network. The Mobile Firewall can block an intruder close to the point of attack. It may be more efficient and more secure.

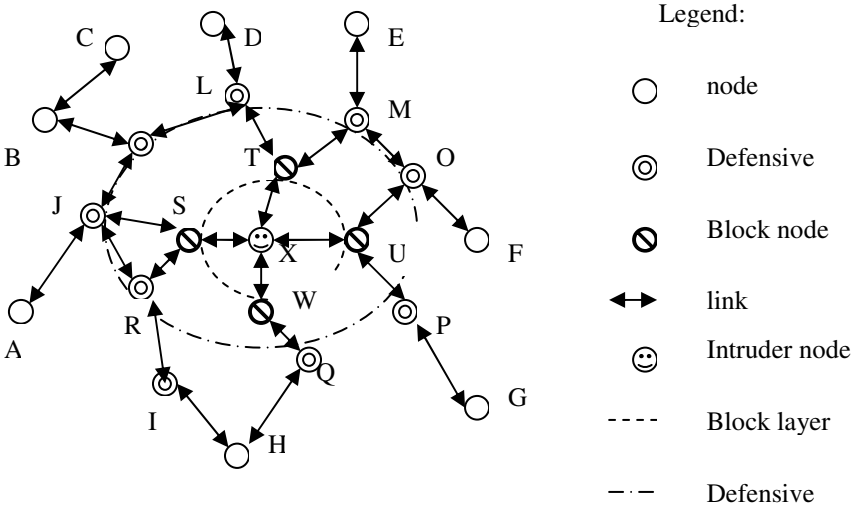


Fig. 2. The Mobile Firewall and intruder

The Mobile Firewall can move with the intruder and isolate the intruder for ever. Because of the dynamic topology of mobile ad hoc network, each node may move with freedom. The intruder may move out of the ring of encirclement of block agents. In order to isolate the intruder at any time, the firewall must be mobile. The Mobile Firewall is composed of two layers, block layer and defensive layer. The block layer is made up of neighbor nodes of intruder, which deter intrusion directly. The defensive layer is made up of the nodes which are outside the block layer.

Figure 2 shows that the Mobile firewall is made of two layers. Mobile ad hoc network is composed of 23 nodes in Fig 2. Node X is intruder. Nodes S, T, U, W con-

struct the block layer of the Mobile Firewall, and they cut off the link between them and intruder X. Node J,K,M,O,P,Q,I,R construct the defensive layer of the Mobile Firewall. These nodes of the defensive layer are outside of the block layer and they are used to prevent intruder from escaping. They can not take effect when they are not neighbor of the intruder. If the intruder moves, some nodes of defensive layer may become the neighbor of the intruder. Just as figure 3, when the intruder gets at a new location, node K and node L becomes the neighbor the intruder. At that time, node K and node L change from defensive node to block node and cut off their link between them and intruder. Node B,C,D become the defensive node because they is outside of the block layer. Node U,W change from block node to defensive node when they are not neighbor of intruder. Node O,P,Q,I become common nodes. Figure 3 shows that the new Mobile Firewall is formed when the intruder moves.

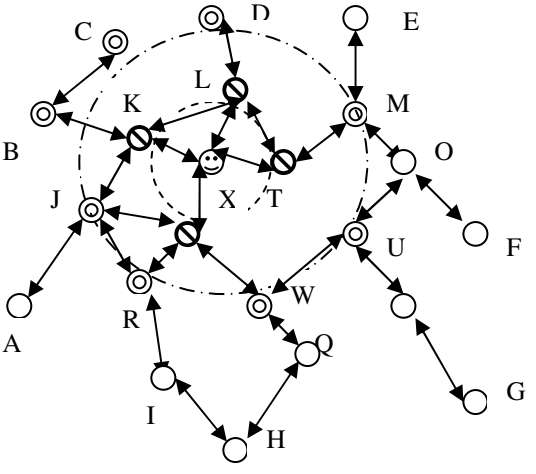


Fig. 3. The Mobile Firewall after intruder moved

In order to save network resource, the mobile firewall need not exist for ever. If the intruder stops attacking, the mobile firewall will terminate after a period. But if intruder continue to attack, the mobile firewall will keep living.

3.3 Shrapnel Way to Dispose

When the intruder is found, the decision agent produces block agents. The block agents get to the neighbour of the intruder and construct the Mobile Firewall to isolate the intruder. There may be some distance between the decision agent and intruder. If the decision agent produces a lot of agents and they all move from the decision agent to intruder, it will spend network bandwidth. In order to save network resource, we present the shrapnel way for agent movement. An artillery shell containing metal balls fused to explode in the air above enemy troops. By analogy, agent moves to the node

around the intruder and reproduces a lot of agents to surround and isolate the intruder in the end.

3.4 Local Repair

When the intruder is found and the Mobile Firewall is built, the routes that include the intruder will break for the intruder can not receive and forward any packet. At that time the process of local repair starts to reroute the path to round the intruder. To repair the path, the node upstream of the intruder broadcasts a Route Discover packet for that destination. If, at the end of the discovery period, the repairing node has not received a Route Reply for that destination, it transmits Route Error to the originating node. The originating node may restart a process of route discover to find a fresh route to the destination. On the other hand, if the node receives one or more Route Reply packets during the discovery period, it first compares the hop count of the new route with the value and chooses a shortest path to replace the current path.

4 Conclusion

This paper presents the architecture of automated intrusion response which is composed of multi-agent. Monitor agents collect the information of each node and send them to decision agents. Decision agents collect the information from the monitor agent and make a judgment by security policies. If the decision agent can judge that some node is invader, it will produce block agents. These block agents move and dispose at the neighbour nodes of the intruder by shrapnel way. The Mobile Firewall is constructed to surround and isolate the intruder in the end. The Mobile Firewall can transfer when the intruder moves and will expire if the intrusion terminated. The whole response process is automated without manual intervention.

The method of intrusion detection in our architecture is too simple to detect some complex attacks. We are doing some work about intrusion detection based finite state machine [15]. We are going to use Coloured Petri Nets [16] to analyze the routing protocol and propose the method of intrusion detection.

References

1. D. Schnackenberg, H. Holliday, R. Smith, K. Djahandari, and D. Sterne, Cooperative Intrusion Traceback and Response Architecture (CITRA), Proceedings of the Second DARPA Information Survivability Conference and Exposition (DISCEXII), Anaheim, CA, June 2001.
2. D. Schnackenberg, K. Djahandari, and D. Sterne, Infrastructure for Intrusion Detection and Response, Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, SC, January 2000.
3. Dan Sterne, Kelly Djahandari, Brett Wilson, Bill Babson,Dan Schnackenberg, Harley Holliday, and Travis Reid, Autonomic Response to Distributed Denial of Service Attacks, In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, RAID 2001, Davis, CA, USA, October 2001, Springer-Verlag, pp.134-149

4. X. Wang, D. Reeves, and S. F. Wu, Tracing-Based Active Intrusion Response, Journal of Information Warfare, Vol. 1, No. 1, September, 2001
5. N. Foukia, S. Hassas, S. Fenet and J. Hulaas, An Intrusion Response Scheme: Tracking the Source Using the Stigmergy Paradigm, in proceedings of Security of Mobile Multi-agent Systems Workshop (SEMAS-2002), Bologna, Italy, July 16 2002
6. CA Carver, JMD Hill, JR Surdu, and UW Pooch, A Methodology for Using Intelligent Agents to Provide Automated Intrusion Response, proceedings of the 2000 IEEE workshop on Information Assurance and Security, West Point, NY,USA, June 2000.
7. Jansen, W. A. Karygiannis, T. Marks, D. G. Mell, P. M., Mobile Agents in Intrusion Detection and Response, Canadian Information Technology Security Symposium , June 19-23,2000 , Ottawa, Canada
8. Srdjan Capkun, Levente Nuttyan, Jean-Pierre Hubaux, Self-organized public-key Management for mobile ad hoc networks, IEEE Transactions on mobile computing, Vol.2, No.1, January-March, 2003
9. Lidong Zhou, Zygmunt J. Haas, Securing ad hoc networks, IEEE Networks Special Issue on Network Security, November/December, 1999
10. P.Papadimitratos, Z.Haas, Secure routing for mobile ad hoc networks, in Proceedings of the SCS communication Networks and Distributed Systems Modeling and Simulation Conference, San Antonio, TX, January 27-31,2002
11. Yih-Chun Hu, Adrian Perrig, David B. Johnson. Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks, in Proceedings of the MobiCom 2002, September 23-28, 2002, Atlanta, Georgia, USA
12. Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, Elizabeth M. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks, In Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP), November 2002
13. Yih-Chun Hu, David B. Johnson, and Adrian Perrig, SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks, in Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), pp. 3-13, IEEE, Calicoon, NY, June 2002
14. Yongguang Zhang & Wenke Lee, Intrusion Detection Techniques for Mobile Wireless Networks, Mobile Networks and Applications, 2003
15. Ping Yi, Yichuan Jiang, Yiping Zhong, Shiyong Zhang, Distributed Intrusion Detection for mobile ad hoc networks, The 2005 International Symposium on Applications and the Internet(SAINT2005), Trento, Italy, January 31 - February 4, 2005
16. S. Gordon, L. Kristensen and J. Billington. Verification of a Revised WAP Wireless Transaction Protocol. In *Application and Theory of Petri Nets 2002*, vol. LNCS 2360, pages 182-202, 2002. Springer Verlag, Berlin Heidelberg New York. ISBN 3-540-43787-8. Presented at the 23rd International Conference on Application and Theory of Petri Nets, Adelaide, Australia, 24-30 Jun 2002.

A Vertical Handoff Decision Process and Algorithm Based on Context Information in CDMA-WLAN Interworking[†]

Jang-Sub Kim, Min-Young Chung, and Dong-Ryeol Shin

School of Information and Communication Engineering,
Sungkyunkwan University,
300 ChunChun-Dong, JangAn-Gu, Suwon, Korea
{jangsub, mychung, drshin}@ece.skku.ac.kr

Abstract. The integration of WLANs and CDMA networks has recently evolved into a very hot issue. In order to support a vertical handoff, we propose a context based vertical handoff decision process and the corresponding algorithms from WLAN to CDMA system, and vice versa, based on wireless channel assignment. We focuses on the handoff decision which uses context information such as dropping probability, blocking probability, GoS (Grade of Service), the number of handoff attempts and velocity. As a decision criterion, velocity threshold is determined to optimize the system performance. The optimal velocity threshold is adjusted to assign available channels to the mobile stations with various handoff strategies. The proposed scheme is validated using computer simulation. Also, the overflow traffic (a vertical handoff) is evaluated and compared with non-overflow traffics in terms of GoS.

1 Introduction

There has been a huge development in wireless communication technologies: mobile technology and WLAN technology. Mobile technologies such as GSM (Global System for Mobile Communications), GPRS (General Packet Radio Service), UMTS (Universal Mobile Telecommunication System) and CDMA (IS-95 A/B and cdma2000) offer high mobility but with low rates. In contrast, WLAN technologies offer high rates but with low mobility. The integration of mobile technology and WLAN technology, which compensates the coverage, bandwidth, and mobility to each other, achieves the requirements of the increasing user demands. In order to provide a convenient access of both technologies in different environments, interworking [1] and integration [2] of the two networks are regarded as a very important work.

Recently, the 3rd generation partnership project (3GPP), a standard body that develops and maintains GSM, GPRS, and UMTS, initiates the specification of interworking architecture for WLANs and 3GPP system. In [3], six interworking scenarios have been identified under different supporting services and operational capabilities.

[†] This work was supported by Korea Science and Engineering Foundation. (KOSEF-R01-2004-000-10755-0).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 601 – 609, 2005.
© Springer-Verlag Berlin Heidelberg 2005

The combination of WLAN and CDMA technology uses the best features of both systems. The key goal of this integration is to develop heterogeneous mobile data network, capable to support ubiquitous data services with very high data rates in hotspots. The effort to develop such heterogeneous networks, especially seamless roaming, is linked with many technical challenges including seamless vertical handoff across WLAN and CDMA technologies, security, common authentication, unified accounting & billing, WLAN sharing, consistent QoS and service provisioning, etc [4].

A handoff mechanism in an overlay CDMA and underlay WLAN should perform well so that the users attached to the CDMA just easily check the availability of the underlay WLAN. The decision criteria for vertical handoff (or network selection) can be based on the maximum link speed, reliability, power utilization, billing, cost, user preference, mobile speed, and Quality of Service like bandwidth, delay, jitter, and loss rate, etc [5]. For simplicity, we do only consider the mobile speed in this paper. A good handoff algorithm is to be derived in order to minimize unnecessary handoff attempts. An appropriate handoff control is also an important issue in system management for the sake of the benefits above with the overlaid cell structures. This paper suggests three handoff strategies : ① no-overflow, ② Overflow – From WLAN to CDMA system, ③ Overflow – From WLAN to CDMA system and vice versa.

In this paper, we deal with a vertical handoff decision process and algorithms based on context information (GoS) and we first propose a context based vertical handoff decision process and the corresponding mechanism between WLAN and CDMA system, based on wireless channel assignment. Secondly, we present a handoff control scheme for a hierarchical structured network. As a decision criterion, velocity threshold is determined to optimize the system performance. The proposed scheme is validated using computer simulation. Also, the overflow traffic (vertical handoff) is evaluated and compared with non-overflow traffics in terms of GoS. The simulation results show that overflows strategy performs as good as other handoff strategy.

The rest of the paper is organized as follows. In Section 2, the proposed vertical handoff algorithms are presented, problems are formulated, and core part of algorithmic decision procedure for the optimal velocity threshold for the WLAN and CDMA selection schemes. Section 3 explains the architecture for integrated networks, the mobility model, performance parameters (i.e. new call blocking probability and handoff call dropping probability, and Grade of Service). Simulations are performed in Section 4 to validate the proposed approach. Finally, the summary of the result and the future related research topics are presented in the conclusion section.

2 A Vertical Handoff Decision Process and Algorithm

A vertical handoff decision process decides when to invoke a vertical handoff operation. The decision process evaluates user location changes (as users may leave or enter a particular network coverage) and context information (QoS, GoS, mobile speed, network preferences, and etc.) of the current and alternative networks. The evaluation of user location changes is carried out based on the Received Signal Strength (RSS). The vertical handoff process is rule based and the rules are decide whether handoff is necessary and to which network. The latter is decided by the GoS

based network selection process invoked when the GoS of a integrated network is below perceived acceptance quality, or GoS is minimized.

Decision rules are described as call initialization, moving out of networks, and entering new networks. Our proposed vertical handoff algorithm between WLAN and CDMA networks considered velocity threshold related to GoS performance and handoff rates is shown in Fig. 1.

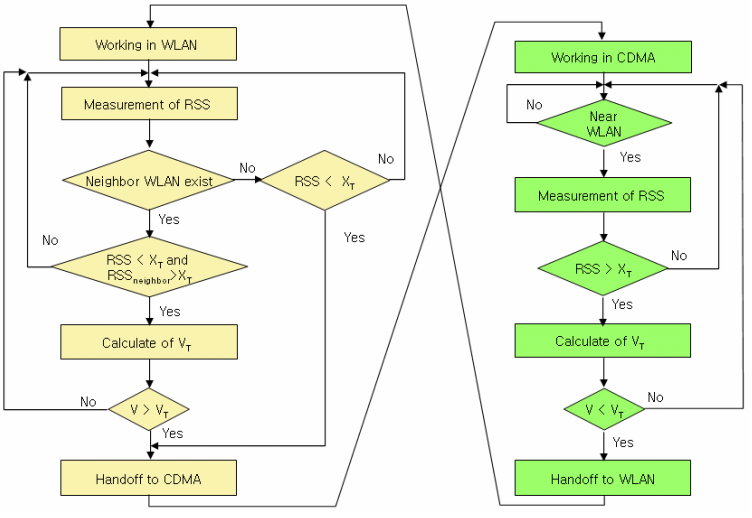


Fig. 1. A vertical handoff algorithm

We use the following variables to determine the vertical handoff:

- X_T : predefined threshold value when the handoff
- V_T : velocity threshold whether a fast mobile station (MS) or a slow MS

When the signal from the WLAN access point (AP) is strong, the MS is connected to the WLAN when the MS is larger than velocity threshold (V_T). As the MS moves away from the coverage of the access point, the signal strength falls. The MS then scans the air for other access points. If no other access point is available, or if the signal strengths from available access points are not strong enough, the handoff algorithm uses this information along with other possible information to make a decision on handing off to the CDMA network.

In a proposed vertical handoff algorithm, the estimation of the velocity threshold procedure is shown in Fig. 2. For the estimation of the mobile speed, Global Positioning System (GPS) or Differential GPS can provide adequate location information. Using GPS and Time-of-Arrival (TOA) information from the user signal, we can estimate for user's velocity. We develop the handoff algorithm based on an optimal velocity threshold. The problem here is to find V_T improving the GoS and decrease the number of handoff attempts (N_h) with the given traffic parameters and MS mo-

bility; $f_{\Lambda}(\lambda)$ and $f_V(v)$. We have to find the velocity threshold satisfied the following equation.

$$\min_{V_T}\{GoS(V), N_h(V)\} \tag{1}$$

The procedure is now concerned with the GoS in which the system wide new call blocking probability PB and the handoff call dropping probability PD are weighted to be averaged as in Equation (11). The GoS can be written as a function of V_T, and hence finding the optimum value of V_T minimizing the value of GoS and N_h is a typical minimization problem.

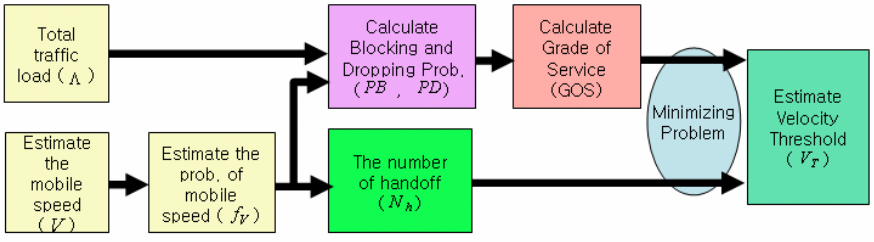


Fig. 2. The estimation of velocity threshold

3 Performance Measures and Analysis

3.1 System Description

We consider a large geographical area covered by contiguous WLANs. Figure 3 shows traffic flows between different wireless networks with related parameters. All the WLANs are overlaid by a large CDMA system. The overlaying CDMA system forms the upper cell layer. Each CDMA system is allocated c_0 traffic channels, and the number of channels allocated to the WLAN cell-i is c_i, $i = 1,2,\cdots,N$. All channels are shared among new calls and handoff calls. MSs are traversing randomly the coverage area of WLAN and CDMA system.

In this paper, all WLANs of the lower layer are treated equally to simplify the overflow. We present analytical results for the proposed system. As stated, our objective is to focus on simple and tractable mechanism for which analytical results can give an insight into handoff between different networks. According to the velocity threshold, all the mobile users are divided into two groups; slower moving users (λ^S) and fast moving users (λ^F). In order to determine the value, which is one of the main goals of this study, a few assumptions related to mobility characteristics are made in system.

The assumptions we employ in the mobility models are taken from [6] as cells are circular with radius R, mobiles are uniformly distributed in the system, mobiles making new calls in WLAN move in a straight line with a direction uniformly distributed between $[0,2\pi)$, and mobiles crossing cell boundary enter a neighbor cell with the incident angle θ of distribution: $f(\theta) = 1/2 \times \cos\theta, \quad -\pi/2 < \theta < \pi/2$.

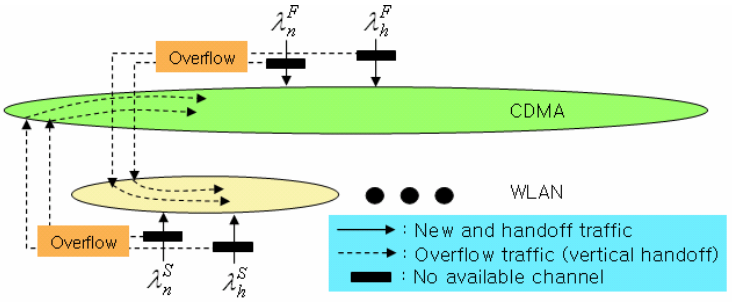


Fig. 3. Management of traffics in integrated system

WLAN cells compose of two types of new call traffics, represented by the call arrival rates λ_n^S and λ_h^S, respectively modeled by the Poisson process (To simplified simulation, voice call considered). Let random variables X and Y denote the straight mobile path for new calls and handoff calls, respectively. With the assumption of the unique WLAN cell size and the same speed of the MS, WLAN cell boundary crossing rate per call (μ_B), provided that no handoff failure occurs [6]: $\mu_B = 2E[V]/\pi R$. New calls assume to finish within the average call duration time, $1/\mu$, or the call handoffs to an adjacent cell. The proportion of the channel returned by the handoff is $P_h = \mu_B /(\mu + \mu_B)$ [7]. In other words, the rate of channel release and that of the call completion due to handoff are $\mu_B /(\mu + \mu_B)$ and $\mu /(\mu + \mu_B)$, respectively.

3.1 The New Call Blocking Probability of WLAN and CDMA System

We denote the blocking probability of calls from CDMA system and WLAN by P_{B0} and P_{B1}, respectively. And the handoff traffic from slow and fast mobiles is denoted as follows. The λ_{h0}^F and λ_{h0}^S is the rate of fast and slow mobile handoff traffic in a CDMA systems, respectively. The λ_{h1}^F and λ_{h1}^S is the rate of fast and slow mobile handoff traffic in a WLAN, respectively. And we denote the take-back traffic rate to CDMA system and WLAN by λ_{T0} and λ_{T1}, respectively. The P_{T0} and P_{T1} are the take-back probability from CDMA system and WLAN, respectively.

The aggregate traffic rate into the WLAN due to a slow MS is computed as follows:

$$\lambda_1^S = \lambda_{n1}^S + \lambda_{h1}^S \tag{2}$$

The aggregate traffic rate into the WLAN due to fast MS is expressed as

$$\lambda_1^F = 1/N \times (\lambda_{n0}^F + \lambda_{h0}^F) P_{B0} + \lambda_{h1}^F \tag{3}$$

The generation rate of the handoff traffic of a slow mobile station in a WLAN is given as follows:

$$\lambda_{h1}^S = P_{h1}^S (\lambda_{n1}^S + \lambda_{h1}^S)(1 - P_{B1}) \tag{4}$$

The generation rate of the handoff traffic of a fast moving MS in a WLAN is characterized as follows:

$$\lambda_{h1}^{F} = P_{h1}^{F}\{1/N \times (\lambda_{n0}^{F} + \lambda_{h0}^{F})P_{B0}(1-P_{B1}) + \lambda_{h1}^{F}(1-P_{B1})\} \tag{5}$$

The parameter ρ is the actual offered load to a WLAN from the new call arrival and the handoff call arrival. Invoking this important property, we can use $\rho_1 = \lambda_1^S / \mu_1^S + \lambda_1^F / \mu_1^F$ as the offered load to the WLAN, the Erlang-B formula calculates the blocking probability with the traffic ρ_1 and the number of channels c_1 [8]

$$P_{B1} = B(c_1, \rho_1) \tag{6}$$

Like as the new call blocking probability of WLAN, we can use $\rho_0 = \lambda_0^S / \mu_0^S + \lambda_0^F / \mu_0^F$ as the offered load to CDMA system, and blocking probability can be written as

$$P_{B0} = B(c_0, \rho_0) \tag{7}$$

3.2 The Handoff Call Dropping Probability of WLAN and CDMA System

Slow MSs are supposed to use WLAN channels. However, since handoff to CDMA system is also allowed, the probability of handoff call drop in WLAN can be calculated as follows. Let P_{10} denote the probability that a slow MS fails to be handoffed to a near WLAN. The probability of the calls, P_{B0}, in a WLAN denotes the probability of failed hand-up to the overlaying CDMA system due to the channel shortage. Then the handoff call dropping probability is

$$P_D^S \approx P_{10}P_{B0} + P_{10}(1-P_{B0})P_{F0}^S \tag{8}$$

Here P_{F0}^S is the probability that a slow MS handoff to CDMA system fail. The P_{10} is defined in such a way that the i th handoff request is successful but the $(i+1)$ th request is dropped:

$$P_{10} = f_1 + s_1 f_1 + s_1^2 f_1 + \cdots = f_1/(1-s_1) \tag{9}$$

where $f_1 = P_{h1}P_{B1}$ and $s_1 = P_{h1}(1-P_{B1})$. f_i describe the probability that handoff fails due to channel shortage and the s_i is the probability of successful handoff. The overall probability of either dropping or handoff failure is

$$PD = R_S P_D^S + R_F P_D^F \tag{10}$$

where R_S and R_F is fraction of slow and fast MSs , respectively

3.3 Grade of Service (GoS)

Among many system performance measures, GoS is most widely used. In fact users complain much more for call dropping than for call blocking. It is evaluated using the prespecified weights, PB and PD ,

$$GoS = (1-\alpha)PB + \alpha PD \tag{11}$$

where PB and PD represent the blocking and dropping prob. of systems, respectively. The weight α emphasizes the dropping effect with the value of larger than one half.

4 Numerical Examples

The proposed procedure is tested with a number of numerical examples for the overlaid structure. The test system consists of 10 WLANs in the CDMA system. The total traffic $\Lambda = \lambda_0 + n\lambda_1$, where λ_0 and λ_1 are the new call arrival rate for the CDMA system and the WLAN, respectively. The radius of the WLAN and the CDMA system are assumed 300m and 1000m, respectively. The average call duration is $1/\mu = 120$ sec. The number of channels in each CDMA system and WLAN is $c_0 = 30$, $c_1 = 10$ for the total $\Lambda = 60$ Erlang. Assume the traffic mobility distribution is same as [6].

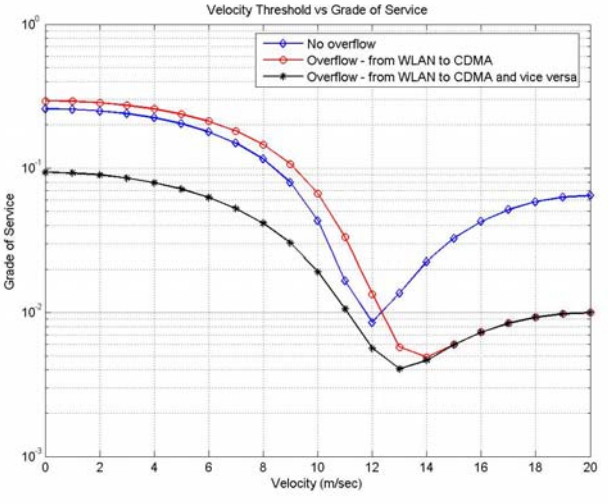


Fig. 4. Grade of Service vs. velocity threshold

In operation phase use can draw a histogram to estimate the $\hat{f}_V(v)$, and the expected value of the mobile speed can be calculated by averaging the mobile speeds monitored by the system. Analytically we can obtain $E[V]$ for such a simple hypothetical velocity distribution [7]. And we consider four handoff strategies for comparison as follows.

① No overflow : A reference system where the two layers are kept completely independent.

② Overflow – From WLAN to CDMA system : A system where only overflow of new and handoff traffic for a slow MS to the CDMA system is allowed.

③ Overflow – From WLAN to CDMA system and vice versa : A system where overflow of new and handoff traffic for both slow and fast MS is allowed.

We investigate the GoS, which is a function of both the traffic load and mobility distribution. Fig. 4 shows the plot of (11) for the mobility distributions of the MS in the system. The vertical arrows in the figure show the range of the possible velocity thresholds at a certain load level. The lowest point in the range corresponds to the maximum allowable and optimal velocity threshold. Optimal V_T is 12m/sec, 14m/sec, 13m/sec for case ①, ②, ③, respectively. Here the GoS of case ② and ③ have minimums of nearly equal values, but V_T does different cases. Case ③ is favorable (See Fig. 4) since V_T in the case ③ is smaller than that of case ② and thus more users are serviced in the CDMA system while the WLAN serves the fewer users. As a result, the WLAN will give rise to a higher number of handoff requests for high-mobility users, and the corresponding number of handoff requests of the calls in progress may cause an excessive processing load in the network. As the velocity threshold increases, the number of handoff attempts in the system also increases.

The overflow strategy, case ③, provides the value of GoS nearly equal to case ② while it has the optimal velocity threshold smaller than that of case ②. With all the observations in mind, the strategy we proposed has desirable characteristics, i.e., finding the optimal value of GoS and the number of handoff rate.

5 Conclusion

We have proposed a vertical handoff decision process with network selection deciding the optimal velocity threshold in order to improve the GoS and minimize the number of handoff attempts with the given traffic volume and three handoff strategies in WLAN and CDMA system. The simulation results show the dependency of the system performance upon the velocity threshold, V_T . The velocity threshold has shown to be an important system parameter that the system provider should determine to produce better GoS and lower handoff rate. From the simulation results we were able to validate the procedures determining the optimal V_T in which depends upon GoS as well as the number of handoff attempts. Furthermore, the overflow strategy (case ③) is more favorable than other handoff strategies in this simulation environment.

References

1. K. Ahmavaara, H. Haverinen, and R. Pichna, “Interworking architecture between 3GPP and WLAN systems,” IEEE Commun. Mag., vol.41, no.11, pp.74-81, Nov. 2003.
2. “Requirements and architectures for interworking between HIPERLAN/3 and 3rd generation cellular system,” Tech. rep. ETSI TR191.957, Aug. 2001.

3. 3GPP TR 22.934, "Feasibility study on 3GPP system to wireless local are network (WLAN) networking," v.1.0.0.0, Release 6, Feb. 2002.
4. Milind M. Buddhikot, Girish Chandranmenon, etal , Design and Implementation of a WLAN/CDMA2000 Interworking Architecture, IEEE Communications Magazine, November 2003.
5. Balasubramaniam, S., Indulska, J., Vertical Handover Supporting Pervasive Computing in Future Wireless Networks, Computer Communication Journal, Special Issue on 4G/Future Wireless networks. Vol 27/8, pp.708-719., 2003.
6. Kwan L. Yeung and Sanjiv Nanda. "Channel Management in Microcell/Macrocell Cellular Radio Systems." IEEE Transactions on Vehicular Technologies, 45(4):601–612, November 1996.
7. JangSub Kim, WooGon Chung, HyungJin Choi and JongMin Cheong, Soon Park, "Determining Velocity Threshold for Handoff Control in Hierachically Structured Networks," PIMRC 98, 1998.
8. W. Fischer and K. S. Meier-Hellstern. "The Markov Modulated Poisson Process (MMPP) cookbook." Performance Evaluations, 18:149–171, 1992.

Dynamic Data Driven Applications Systems: New Capabilities for Application Simulations and Measurements

Frederica Darema

National Science Foundation, Arlington VA 22230, USA
darema@nsf.gov

Abstract. The Dynamic Data Driven Application Systems (DDDAS) concept entails the ability to incorporate dynamically data into an executing application simulation, and in reverse, the ability of applications to dynamically steer measurement processes. Such dynamic data inputs can be acquired in real-time on-line or they can be archival data. DDDAS offers the promise of improving modeling methods, augmenting the analysis and prediction capabilities of application simulations, improving the efficiency of simulations and the effectiveness of measurement systems.

In the recent years grid computing technologies provide advanced computational capabilities for applications and application simulations At the same time measurement infrastructures, from instruments to sensor systems, to data storage technologies and remote data access have also matured. The DDDAS concept dynamically integrates computational and measurement aspects of an application, and in that respect the "platform" supporting the application becomes a unified computational and measurement infrastructure. Enabling the synergistic feedback and control-loop between application simulations and measurements requires novel application modeling approaches and interfaces to measurement systems, mathematical and statistical algorithms tolerant to perturbations from dynamic data inputs, and systems software to support the dynamic resource requirements of such applications. This and the rest of the papers in the proceedings of this workshop provide examples of ongoing research developing DDDAS technologies, both in terms of new capabilities in applications, measurement methods and in related systems software technologies.

1 Introduction

The Dynamic Data Driven Application Systems (DDDAS) [1,2] concept entails the ability to incorporate dynamically data into an executing application simulation, and in reverse, the ability of applications to dynamically steer measurement processes. Such dynamic data inputs can be acquired in real-time on-line or they can be archival data. The DDDAS concept offers the promise of improving modeling methods, augmenting the analysis and prediction capabilities of application simulations, improving the efficiency of simulations and the effectiveness of measurement systems.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 610 – 615, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Traditional applications and systems simulations, thought fairly complex, are still lacking the ability to accurately describe such systems. This situation is accentuated in cases where real-time dynamic conditions exist. Application simulations that can dynamically incorporate at runtime new data, archival or from on-line measurements of the actual systems, offer the promise of more accurate analysis, more accurate predictions, more precise controls, and more reliable outcomes. These capabilities are fostered with the DDDAS concept. DDDAS can be viewed as a methodology to counterbalance incompleteness in model and capability to enhance the application models by imparting additional information into the model as at runtime additional data are used to selectively enhance or refine the original model. It can also be used to reduce the time to derive the simulation results, an important capability in many simulations, including simulations of real-time events.

In addition the ability of an application simulation to control and guide the measurement processes, creates more effective measurement processes. Such capabilities are highly desirable in cases where measurements are difficult to perform, time-critical, or expensive, in that can reduce collection time, cost, improve the quality of data collected.

Thus the concept of DDDAS goes beyond the current notions of "systems of systems" which primarily refer to multi-modal computational aspects of an application. DDDAS extends the concept of the "systems of systems' to also include the "measurement" as one of the systems.

In the recent years technologies to support the computational aspects of applications and application simulations have considerably advanced, as exemplified by the advent of grid computing technologies and computational grid infrastructures. At the same time measurement infrastructures, from instruments to sensor systems, to data storage technologies and remote data access have also matured. More systematic infrastructures for computing platforms and for measurement systems have been also recently called for [4]. The DDDAS concept dynamically integrates computational and measurement aspects of an application, and in that respect the "platform" supporting the application becomes the unified computational and measurement infrastructure. This kind of integrated, unified infrastructure goes beyond the current notions of computational grids; one could call it the "extended grid" or the integral grid.

To enable the DDDAS paradigm requires synergistic multidisciplinary research projects. Research in such projects needs to address in an integral way application modeling and simulation methods, mathematical and statistical algorithms, measurement methods, and systems software to support DDDAS environments. Opportunities in enabling DDDAS and challenges in creating DDDAS capabilities, have been discussed in [1,2,3] and elsewhere. A brief list of such challenges includes: advances at the applications level for enabling this kind of dynamic feedback and coupling with measurements; advances in complex, multi-modal models, and ability to combine dynamically such models as is dictated by the injected data, advances in the applications algorithms for the algorithms to be amenable to perturbations by the dynamic data inputs and enhanced capabilities for handling uncertainties in input data; new technology in the computer software systems areas to support such environments where the applications requirements are dynamic, depending on the dynamic data inputs. Furthermore the DDDAS concept by integrating the computational and measurement aspects of an application, goes beyond the notions of

cyber-infrastructure. Thus many aspects such as reliability, security, fault tolerance, system recovery extend beyond the technologies developed for computational grids, to accommodate the support not only of computational grids but also of systems like sensor networks and other measurement instruments.

The efforts presented here provide examples of technologies and capabilities that are being developed based on the DDDAS concept and supporting DDDAS environments. This workshop is a third of a series of workshops of the DDDAS topic that have been organized as part of ICCS, starting with ICCS'03.

2 Overview of Selected Research Efforts Presented in This Workshop

The rest of the papers presented in this workshop represent ongoing research projects each addressing a number of aspects along the technical challenges mentioned above. With the exception of project [16], all other projects [6-15, 17-19] are predominantly funded by NSF ITR program [5]. Several papers address application simulations whose capabilities are enhanced by dynamic data inputs into the executing application/simulation. While many of the papers are developing new capabilities at the application and algorithm levels, several of the papers focus in the underlying software technology to support the development of the complex DDDAS applications, their dynamic runtime support requirements and interfaces to control measurement systems. Several papers provide examples of simulation controlled measurement processes. Most of the work on mathematical algorithms tolerant and stable to dynamically streamed data is done in the context and together with the development of the application models. In all cases the mathematical algorithms and systems software development is done in the context of specific applications driving and validating the developed technologies.

The projects discussed in [6-11, 21] enable improved analysis and prediction capabilities of application simulations and/or speeding up the computation times together with improved quality of simulations, and in a feedback loop the simulations control of measurements, sensors and other instruments. The applications range from structural analysis, environmental and natural resource management, and crisis-management. In [6] the project employs sensor originating data-driven pre-computed solutions synthesis to speedup simulation time and fidelity of result for applications that have real-time constraints such as supersonic platforms and fire dynamics simulations. In [7] employs dense networks of high resolution phased array and doppler radars to adaptively respond to weather patterns and produce better-than-real-time predictions of tornadoes and other "meso-scale" weather events by simulations and measurements continually steering by new data, optimizing data collection, steering remote observing and selectively changing configuration of measurement instruments. The paper in [8] develops capabilities for short-range forecasting of wildfire behavior from real-time weather data, thermal imaging, and thermal sensor data streams. In [9] algorithms to incorporate measurements in algorithms for underground contaminant simulations, and in [10] efficient algorithms to dynamically integrate observational data with models for improved forward and backward (adjoint) atmospheric chemical transport modeling. The project in [11] discusses

development of application models and algorithms, as well as a systems software infrastructure for optimization of the process of secondary oil recovery. The project in [21] has developed high-fidelity simulations of large structures, subject to mechanical and thermal stesses.

Applications in [12-14] are in the areas of medical and biological sciences. The project in [12] uses the DDDAS concept to enable real-time intra-operative image guided neurosurgery, by developing applications and the underlying infrastructure to support such environments. In [13] spectroscopic data are dynamically input into protein dynamics computations speeding-up and improving the fidelity of the computations. [14] develops models to simulate the air flow around bat wings during flight.

The projects in [15-17] develop multimodal simulation models and integration of such multimodal models. The project in [15] develops simulations for decision support systems, such as case scenario for evacuation of a building in case of fire and structural damage to a building due to an explosion. In such "system of systems" simulation scenario, numerical PDE and FEM models simulating fire dynamics and structural aspects of the building are integrated w agent based models that simulate behavior of people in a panic situation, evacuation policies, and other factors. The models are coupled and also driven by dynamic data inputs from sensors monitoring smoke and temperature increases due to the fire, sensors monitoring air flow, and feeding all such data into the fire model, to other sensors monitoring structural integrity aspects, and feeding that to the structural analysis program, and in turn such models coupled with the agent based evacuation simulation models. Under the same project simulations are developed when not only a building but a section of a city is affected. The project in [16] is using the DDDAS concept, to develop a decision support system for emergency medical services. One can envision the scenario, where the capabilities in [15] are further coupled with those in [16] to develop an end-to-end decision support system of how to evacuate and optimize medical services in case of an adverse attack to a building or a city section. Projects [15] and [16] coupled with [21,] which is simulating the crash of an aircraft onto a building, could be used to fully simulate the scenario of a crisis situation caused by such an adverse event. The project in [17] uses the DDDAS concept to couple continuous and discrete modeling methods for 3D tracking, and overcoming the shortcomings of each of these two methods when applied individually. Improvements in such methods can enable computer-based sign-language analysis, and other human-computer interaction applications such, human-identification and computer-based recognition of human deception.

The projects in [18-20] address systems issues and other fundamental aspects of DDDAS systems, such as: ability to invoke dynamically application modalities as dictated by the dynamic data inputs [18], runtime environments to support the fault-tolerant underlying infrastructure required by DDDAS systems [19], and aspects of how such "systems-of-systems" environments, where dynamic multimodal models and measurement systems are integrated can be validated.

3 Summary

The DDDAS concept opens new capabilities in applications and measurements. Through efforts that have started developing DDDAS systems, that promise begins to

materialize. While we are far from having fully and robustly developed these capabilities and the underlying technologies, we are in a steady path towards accomplishing that objective. The path is long, and the progress made is exemplified by the projects presented in the ICCS/DDDAS workshops, including the present one.

References

1. NSF Workshop, March 2000; www.cise.nsf.gov/dddas
2. F. Darema, Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements, ICCS'04
3. F. Darema, Grid Computing and Beyond: The Context of Dynamic Data Driven Applications Systems, Proceedings of the IEEE, Special Issue on Grid Computing (in 3/05)
4. Cyberinfrastructure Report http://www.communitytechnology.org/nsf_ci_report
5. NSF Information Technology Research (ITR) Program (1999-2004)
6. J. Michopoulos, C. Farhat, E. Houstis, P. Tsompanopoulou, H. Zhang, and T. Gullaud, Dynamic Data Driven Methodologies for Multiphysics System Modeling and Simulation; Proceedings ICCS'05
7. Beth Plale, Dennis Gannon, Dan Reed, Sara Graves, Kelvin Droegemeier, Bob Wilhelmson, and Mohan Ramamurthy, Towards Dynamically Adaptive Weather Analysis and Forecasting in EAD; Proceedings ICCS'05
8. Jan Mandel, Lynn S. Bennethum, Mingshi Chen, Leopoldo P. Franca, Craig J. Johns, Janice L. Coen, Craig C. Douglas, Minjeong Kim, Andrew V. Knyazev, Robert Kremens, Vaibhav Kulkarni, Guan Qin, Anthony Vodacek, Jianjia Wu, Wei Zhao and Adam Zornes, Towards a Dynamic Data Driven Application System for Wildfire Simulation; Proceedings ICCS'05
9. Craig C. Douglas, Yalchin Efendiev, Richard Ewing, Victor Ginting, Raytcho Lazarov, Martin J. Cole, Greg Jones and Chris R. Johnson, Multiscale Interpolation, Backward in Time Error Analysis for Data-Driven Contaminant Simulation; Proceedings ICCS'05
10. Adrian Sandu, Emil M. Constantinescu, Wenyuan Liao, Gregory R. Carmichael, Tianfeng Chai, John H. Seinfeld, and Dacian Daescu, Ensemble-Based Data Assimilation for Atmospheric Chemical Transport Models; Proceedings ICCS'05
11. Manish Parashar, Vincent Matossian, Wofgang Bangerth, Hector Klie, Benjamin Rutt, Tahsin Kurc, Umit Catalyurek, Joel Saltz, and Mary F. Wheeler, Towards Dynamic Data-Driven Optimization of Oil well Placement; Proceedings ICCS'05
12. Amit Majumdar, Adam Birnbaum, Dong Ju Choi, Abhishek Trivedi, Simon K. Warfield, Kim Baldridge, and Petr Krysl, A Dynamic Data Driven Grid System for Intra-operative Image Guided Neurosurgery; Proceedings ICCS'05
13. Dimitrios Morikis, Christodoulos A. Floudas, and John D. Lambris, Structure-based Integrative Computation and Experimental Approach for the Optimization of Drug Design; Proceedings ICCS'05
14. I.V. Pivkin, E. Hueso, R. Weinstein, D. H. Laidlaw, S. Swartz, and G. E. Karniadakis, Simulation and Visualization of Air Flow Around Bat Wings During Flight; Proceedings ICCS'05
15. R. Chaturvedi, S. A. Filatyev, J.P. Gore, A. Hanna, J. Means, and A.K. Mellema,, Integrating Fire, Structure and Agent Models; Proceedings ICCS'05
16. Mark Gaynor, Margo Seltzer, Steve Moulton, and Jim Freedman,, A Dynamic, Data-Driven, Decision Support System for Emergency Medical Services; Proceedings ICCS'05

17. Dimitris Metaxas and Gabriel Tsechpenakis, Dynamic data Driven Coupling of Continuous and Discrete Methods in 3D Tracking; Proceedings ICCS'05
18. David Brogan, Paul Reynolds, Robert Bartholet, Joseph Carnahan, and Yannick Loitiere,, Semi-Automated Simulation Transformation for DDDAS; Proceedings ICCS'05
19. Andrew Grimsahw, Marty Humphrey, John C. Knight, Anh Nguyen-Tuong, Jonathan Rowanhill, Glenn Wasson, and Jim Basney, The Development of Dependable and Survivable Grids; Proceedings ICCS'05
20. J. Michopoulos and Sam Lambrakos, On the Fundamental Tautology of Validating Data-Driven Models and Simulations; Proceedings ICCS'05
21. Christoph Hoffman, Ahmed Sameh, Ananth Grama, High-Fidelity Simulation of Large Scale Structures; Proceedings ICCS'05

Dynamic Data Driven Methodologies for Multiphysics System Modeling and Simulation

J. Michopoulos[1], C. Farhat[2], E. Houstis[3,4], P. Tsompanopoulou[4], H. Zhang[3], and T. Gullaud[2]

[1] Code 6390, Special Projects Group, U.S. Naval Research Laboratory, U.S.A
john.michopoulos@nrl.navy.mil
[2] Dept. of Mechanical Engineering, Stanford University, U.S.A
{cfarhat, tgullaud}@stanford.edu
[3] Computer Sciences Dept. & Electrical and Computer Engineering Dept., Purdue University, U.S.A
{enh, hzhang}@purdue.edu
[4] Dept. of Comp. Eng. and Telecommunications, University of Thessaly, Greece
{enh, yota}@uth.gr

Abstract. We are presenting a progress overview associated with our work on a data-driven environment for multiphysics applications (DDEMA). In this paper, we emphasize the dynamic-data-driven adaptive modeling and simulation aspects. Adaptive simulation examples of sensor-originating data-driven precomputed solution synthesis are given for two applications. Finally, some of the computational implementation details are presented.

1 Introduction

Our project has been focusing on developing methodologies and corresponding technologies, for utilizing dynamic data to generate adaptive behavioral models of physical systems as well as facilitate their realistic and near real time behavior simulation. Of particular interest here are systems that are excited by multi-field conditions and often require multi-domain modeling from a continuum systems perspective. Characteristic examples of such systems are fire dynamics simulation as well as material softening trends as a function of flight conditions of a supersonic aircraft for prediction of future behavior as detected by sensor originating data-streams.

This paper presents an overview of some of the recent progress associated with our project on adaptive modeling and simulation for supersonic aircraft response prediction for structural design, material qualification/certification needs, and fire dynamics propagation applications relevant to crisis management and decision support for first responders. It also describes the predominant features of the computational implementations. Previous progress on the project can be found elsewhere [1, 2, 3].

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 616–623, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Adaptive Constitutive Modeling of Composites

The concept of "data-driven system modeling" has been frequently identified by the majority of research community as the process of system identification. Various techniques exist to utilize data representing discrete values of systemic behavioral parameters (inputs/outputs) as they occur from exercising the physical system (i.e. through systematic experimentation or real time monitoring) to determine free coefficients/constants involved in the analytical (i.e. mathematical) or/and computational model of the physical system at hand. Design optimization involving objective function minimization is arguably the most popular among these techniques.

The concept of "dynamic-data-driven modeling" is an extension of the "data-driven modeling" in that it attempts to identify a system that may or may not be changing itself at real-time. In this case, the model is being formed or determined just prior to being used based on the dynamic data.

In the context of the systems associated with DDEMA we are interested in adaptively and dynamically constructing the appropriate models. We have identified two general meanings behind the term "adaptive modeling" as it has been used by various investigators. The most frequent one refers to selecting a fixed analytical representation for the model and then use the incoming dynamic data to determine the free coefficients and/or parameters associated with it. It is anticipated that as more and more data are coming in the model's identification process the model itself is adapting through the re-computation of the unknown coefficients/parameters till the data are exhausted or the model reaches its fixed point (if it exists). We call this type of adaptive modeling the "static in-line modeling" (SILM) type. However, there is a possibility that the system changes its behavior very drastically and the SILM approach does not reach fixed point nor it can reproduce older data but only recent data. This is the case for strongly non-linear systems that can be approximated by various models that can be captured by either physics aware or physics-agnostic representations [7]. This case calls for the existence of a meta-level modeling infrastructure that can select representations adapted to the data and will be designated as "dynamic meta modeling" (DMM). Clearly for each area of applicability DMM selects a model that still has to be identified by SILM. DMM is not a widely known approach especially because its computational and implementation cost and complexity.

We have initiated the extension of NRL's traditional SILM approach for characterizing or identifying the multi-field constitutive response of composites under mechanical and heat flux loading, to a DMM approach by appropriate modification of our mechatronic automation methodology [7]. Composite material characterization under thermal and mechanical loading is essential for the application area where DDEMA will be validated for the case of aircraft modeling activity. The appropriate design scenario involves mechanical pressure and heat fluxes that are applied on aircraft skins due to the structure-fluid interaction induced by supersonic flight conditions. The specific approach is based in extending the notion of dissipated energy density (DED). DED is the function that is traditionally constructed as a sum of basis functions weighted by

unknown coefficients to be determined trough the identification process from mechatronically derived experimental data [7]. These coefficients represent the material model parameters and the procedure to identify them has always assumed isothermal and adiabatic conditions for the material under identification. However, the co-existence of aerodynamically induced heat fluxes has forced reconsideration of the isothermal and adiabatic assumptions for the identification process. There are multiple ways to account for a material that represents an open system that can exchange heat with its environment. Allowing temperature to be a state variable of the DED function of the system in addition to the strain components, is one of them. Another possible extension is to allow the parameters of the isothermal model to become functions of temperature themselves. Many other extensions can be constructed by considering varying combinations of both of the previous extensions. These extensions constitute the template domain for the DMM approach. A meta-optimization scheme outside each individual SLIM can select the best one of all these extensions and is the embodiment of the DMM methodology that will be further developed.

3 Dynamic Data Driven Adaptive Simulation

To enable and improve dynamic data driven adaptivity of systemic behavioral simulation with real-time performance, the analytical and computational complexity of the models involved requires that techniques and methodologies are developed and employed to accelerate computation. It is essential that incoming data originating from sensors associated with the actual physical systems are utilized to exercise their models in a manner that achieves this objective.

An extended description of the techniques available for reducing model complexity through reduced order modeling, and for reducing total computation time by implementing combined computational parallelism based in domain decomposition techniques in both space and time has been given elsewhere [4, 5, 6].

Here we are focusing on sensor-originating data-driven solution synthesis from pre-existing basis cases behavioral simulations.

Let r_i represent a scalar field corresponding to a response component of the behavior of a continuous system indexed by the position index i usually corresponding to some nodal identification address as it relates to a mesh used for determining it numerically. The context of this response component includes the existence of some analytical model (i.e. set of partial differential equations), some discretization scheme (i.e. using finite element/volume analysis over an appropriately selected mesh) that when applied yields the values of all output state variables of the system (i.e. temperature, strain components, concentrations etc.) as a column array $\mathbf{r}$ containing its values r_i where $i = 1, \ldots, m$ for all mesh points m. It is assumed that there is a column vector $\mathbf{f}$, of n global input parameters $f_j, j = 1, \ldots, n$, regarded as the forcing or loading inputs into our systemic simulation model in a manner that potentially effects all r_i at every i point. To obtain the base case solutions $\mathbf{b}_j$, we apply the solution scheme for our system's of PDE-based model n times for each one of the forcing inputs by

setting $f_j = 1$ for each forcing function. We further postulate as in [8], that there is an appropriate selection of forcing input parameters such as any response of the system can always be expressed as a linear combination of this base case solutions, according to,

$$\mathbf{r} = \mathbf{B}\mathbf{c}, \tag{1}$$

where $\mathbf{B}$ is the rectangular $n \times m$ array constructed such as $\mathbf{B} = [\mathbf{b}_1(f_1), \cdots, \mathbf{b}_n(f_n)]$ that has elements defined by $B_{ij} = b_i(f_j)$, and $\mathbf{c}$ is an n-dimensional column array containing the proportion coefficients c_j multiplying each one of the base cases to construct the final solution.

It is important to recognize that $\mathbf{B}$ can be computed prior to its utilization from Eq. 1 at the time of of solution synthesis for real-time simulation. In fact, the final behavioral solution synthesis is achieved by the computational implementation of Eq. 1 provided $\mathbf{c}$ is known.

Determination of $\mathbf{c}$ can be achieved via the use of q sensors embedded in the physical system at positions k that can be selected to coincide with some of the points i and they measure the sensor response column vector $\mathbf{s}$ that contains individual sensor responses $s_l, l = 1, \ldots, q$. Since s_l essentially is r_l at the point where a sensor exists, and without loss of generality we assume noiseless sensors, then we can repeat the previously applied argumentation and approximate the sensor response according to

$$\mathbf{s} = \mathbf{Q}\mathbf{c}. \tag{2}$$

$\mathbf{Q}$ represents a limited version of $\mathbf{B}$ such that only the points on the mesh where sensors are located are considered. Thus $\mathbf{Q}$ is an $n \times q$ rectangular array. Clearly for any case where $n \leq q$ (i.e. base cases are less than the number of available sensors), there is a pseudo inverse $\mathbf{Q}^+$ of $\mathbf{Q}$, such that Eq. 2 can be inverted to determine $\mathbf{c}$ and subsequently inserted in Eq. 1 to yield

$$\mathbf{r} = \mathbf{B}\mathbf{Q}^+\mathbf{s}. \tag{3}$$

This relation allows to compose the response of the system in terms of pre-computed and stored solutions (i.e. $\mathbf{B}$ and $\mathbf{Q}$ and their product) and sensor originating data (i.e. $\mathbf{s}$). In this manner, we avoid spending the time for re-applying the PDE solution scheme that involves time-consuming computational application of space and time integration (we only do this for the base cases), and substitute it to much more computationally efficient sequences of multiplications and additions as defined by the computational implementation of the matrix operations designated by Eq. 3. It is also worthwhile mentioning here that the solutions represented by $\mathbf{B}$ may be constructed in terms of reduced order models to further accelerate total computation time for both the pre-computing and real-time stages of the process.

Application 1: Aircraft Behavior Prediction. We have initiated work for applying the pre-computed synthesis approach described for the case of subsonic to supersonic regime of the AGARD wing 445.6 [9]. This wing is an AGARD standard aeroelastic configuration with a 45 degrees quarter-chord sweep angle, a panel aspect ratio of 1.65, a taper ratio of 0.66, and a NACA 65A004 airfoil

section. In our previous work [10] we have modeled and solved the three-field PDE system governing fluid-structure aeroelastic response of this wing and have established favorable comparisons with experimental results. Results for various state variables have been obtained for various flight conditions defined by an appropriate variation of the far-field mach number (denoting the speed of the corresponding aircraft) and angle of attack. These two parameters are also selected as the forcing input parameters for our system.

To apply the above mentioned scheme, we have selected the total strain energy density (SED) per unit of volume, as the behavioral response state variable of choice. In addition, we assumed the presence of 4 sensors that can measure in-plain strains on specific places of the wing. We focused on small region of the parameter space defined by ($M_\infty \in [0.6, 0.8]$, $\alpha \in [6^o, 8^o]$), in order to allow proportional superposition as it is enforced by the composition equations.

Figure 1 shows nine SED distributions of the AGARD wing resulting for 3 values of M_∞ and α respectively. All cases surrounded by dash-lined frames have been both synthesized and computed from the direct solution procedure. Differences between synthesized and direct SED distributions never exceeded 4.2 % root mean square error. The red dots shown in the case of $(M_\infty, \alpha) = (0.6, 6^o)$ represent the location of the sensors.

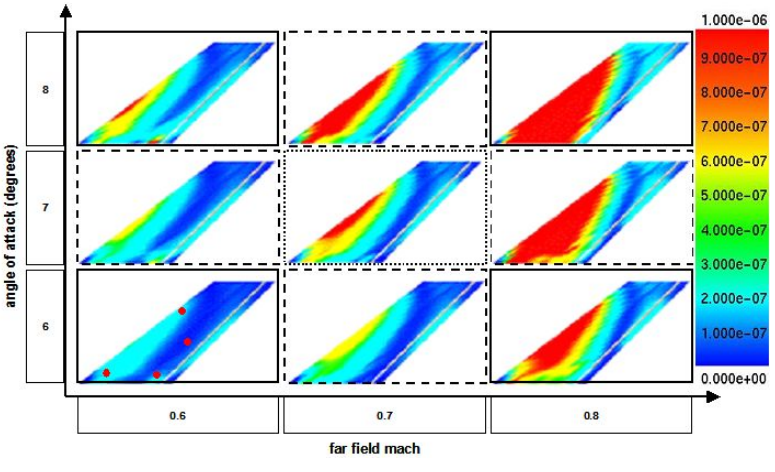


Fig. 1. Strain energy density distributions of the AGARD wing. All distributions along $M_\infty = 0.7$ and $\alpha = 7^o$ have been both computed and reconstructed and the differences are undiscernible

Direct computation of any of these solutions took an average of 19.6 minutes on six processors of a SGI Origin 2000. It is indicative that solution synthesis of the same case implemented as a GAWK script running on a 3.2 GHz Pentium-4 processor takes 0.6 seconds. Thus a realization of 1960 times speedup is achieved (without adjusting for the different processor unit number and architecture).

Application 2: Fire Dynamics Behavior Prediction. Application of the solution synthesis approach based on dynamic sensor data requires the establishment the base case parameter space. The problem of fuel leak at room inside a submarine has been selected to represent the physical context for this case. "Fire Dynamic Simulator" (FDS) /citeFDS01 has been selected to numerically integrate the corresponding reactive flow PDE-model.

We have selected a rectangular 5×3 array of 15 thermocouple sensors that measure temperature. We have chosen the excitation parameters for constructing our base cases to be: total time combustion is ongoing (t_c in seconds); position-X of leak (f_x in meters); position-Y of leak (f_y in meters); fuel's Heat Release Rate Per Unit Area to designate the fire strength (f_s in kW/m^2). They define a four dimensional parameter space and in particular a bounded sub-region defined by $\{t_c \in [0., 10.], f_x \in [0., 4.7], f_x \in [0., 2.5], f_s \in [100., 500.]\}$.

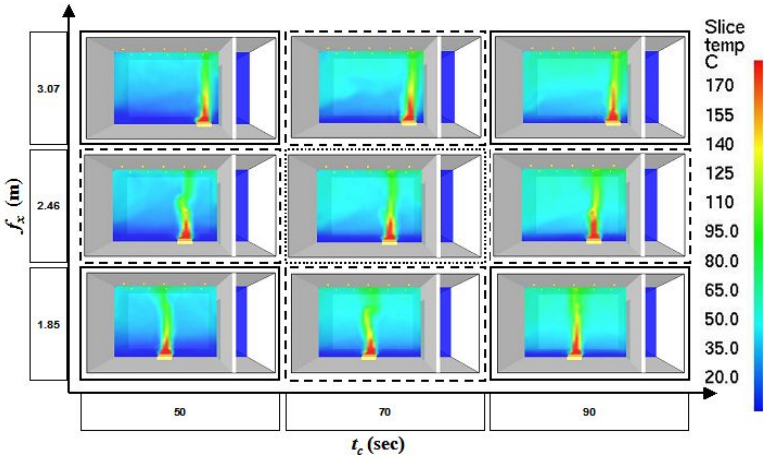


Fig. 2. Temperature distribution along x-axis at the mid section of a submarine room with a fuel leak. All the distributions for $t_c = 70$ sec and $f_x = 2.46$ m have been both computed and reconstructed and the differences are undiscernible

While the average time for all FDS-computed cases was 13.4 minutes on a 3.2 GHz Pentium 4 system, the synthesized solutions took 1.6 seconds on the same system. Therefore the average speedup factor was 817. It should be noted that the synthesized runs in addition to the actual computation include the data input and output time.

4 Computational Implementation Progress

Progress regarding the computational implementation of DDEMA is reported for the cases that follow.

Supersonic Platform Simulation: The core technology behind the aircraft multiphysics simulations for our project is the AERO suite of codes [12]. To provide Operating System and system architecture transparency along with ease of use we have developed a JAVA-based front-end called JAERO. JAERO is a full JAVA stand-alone application that runs on the local client machine in front of the user and provides automated configuration, data integrity, dependency and consistency checking, file management, execution control and monitoring management services to the AERO suite installed on a remote cluster. JAERO does not enforce continuous master-slave relationship to the AERO suite executables and furthermore it provides job monitoring services at will. State storage and recovery has been enabled by utilization of XML-based file encoding, storage and recovery. It does not require HTTPD server presence since it exploits SSH client-server daemon based multi-threaded communications. RSA fingerprints are used for host authentication and encryption has been implemented for all data transmitted over the network. XML scripting of a full runtime session of AERO that encapsulates all the inter-code parametrization, data file location, cluster e.t.c is also provided and it will eventually provide extended availability to 3rd parties. JAERO is currently undergoing beta testing and is be extended to support data visualization post-processing.

Fire Dynamics Simulation: The core technology behind the reactive flow modeling and simulation for fire dynamics prediction applications, is the FDS legacy code. To enable lightweight parallelism capability on distributed heterogeneous networked systems, and Operating System and system architecture transparency along with ease of use, we have developed JAMPI. JAMPI is a set of resources for "J"ava "A"gent enabling of the "M"essage "P"assing "I"nterface for legacy MPI-based codes. In particular, we used JAMPI to develop the Java Fire Dynamic Simulator (JFDS) that resulted in converting the MPI version of FDS into a JADE /cite agent implementation of it. This approach enables instances of legacy codes installed on various shared memory architectures, to appear as light-weight agents accessing the codes directly or their corresponding GRID or WEB services encapsulation. Therefore, all benefits of using agent-based middleware are automatically realizable for legacy codes in general and for FDS in particular.

JAMPI is in alpha stage while JFDS is in beta testing stage and we are currently undergoing through validation procedures along with performance analysis and comparison studies.

Acknowledgments

The authors acknowledge the support by the National Science Foundation under grant ITR-0205663. Partial support from NRL's 6.1 core-funding is also greatly acknowledged.

References

1. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Rice, J., Farhat, C., Lesoinne, M., Lechenault, F., DDEMA: A Data Driven Environment for Multiphysics Applications,in: Proceedings of International Conference of Computational Science - ICCS'03, Sloot, P.M.A., et al. (Eds.) Melbourne Australia, June 2-4, LNCS 2660, Part IV, Springer-Verlag, Haidelberg, (2003), 309-318.
2. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Farhat, C., Lesoinne, M., Rice, J., Joshi, A., On a Data Driven Environment for Multiphysics Applications, Future Generation Computer Systems, in-print, (2005).
3. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Joshi, A., Agent-based Simulation of Data-Driven Fire Propagation Dynamics, Proceedings, International Conference Computational Science - ICCS'04, Krakow, Poland, June 6-9, 2004, Part III Series: Lecture Notes in Computer Science, Bubak, M.; Albada, G.D.v.; Sloot, P.M.A.; Dongarra, J. (Eds.), LNCS 3038, (2004), 732-739.
4. Farhat, C., Chandesris, M., Time-Decomposed Parallel Time-Integrators: Theory and Feasibility Studies for Fluid, Structure, and Fluid-Structure Applications, Internat. J. Numer. Meths., Engrg. 58, (2003), 1397-1434.
5. Lieu, T., Lesoinne, M., Parameter Adaptation of Reduced Order Models for Three-Dimensional Flutter Analysis, 42^{nd} AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, (2004).
6. Michopoulos, J., Farhat, C., Houstis, E., Dynamic-Data-Driven Real-Time Computational Mechanics Environment, Proceedings, International Conference Computational Science - ICCS'04, Krakow, Poland, June 6-9, 2004, Part III Series: Lecture Notes in Computer Science, Bubak, M.; Albada, G.D.v.; Sloot, P.M.A.; Dongarra, J. (Eds.), LNCS 3038, (2004), 693-700.
7. Michopoulos, J.G., Mechatronically Automated Characterization of Material Constitutive Response, in Proceedings of the 6th World Congress on Computational Mechanics (WCCM-VI), September 5-10 2004, Beijing China, Tsinghua University Press and Springer, (2004), 486-491.
8. Michopoulos, J.G., Mast, P.W. , Badaliance, R., Wolock, I., Health Monitoring of smart structures by the use of dissipated energy, ASME proc. 93 WAM on Adaptive structures and material systems, G.P. Carman/E. Garcia, eds., ASME, AD-Vol. 35, (1993), 457-462.
9. Yates, E., AGARD Standard Aeroelastic Configuration for Dynamic Response, Candidate Configuration I. – Wing 445.6, NASA TM-100492, (1987).
10. Michopoulos,J. , Badaliance, R. , Chwastyk, T. , Gause, L., Mast, P., Farhat C. , and Lesoinne M., Material Softening Issues In A Multiphysics Virtual Wind Tunnel Environment, AIAA 2002-1095, proceedings of the 2002 AIAA conference at Reno Nevada January 14-17 2, (2002).
11. McGrattan, K.B., Baum, H.R., Rehm, R.G., Hamins, A., Forney, G.P., Fire Dynamics Simulator - Technical Reference Guide, National Institute of Standards and Technology, Gaithersburg, MD., NISTIR 6467, (2000).
12. Farhat, C., Geuzaine , P., Brown, G., Application of a Three-Field Nonlinear Fluid-Structure Formulation to the Prediction of the Aeroelastic Parameters of an F-16 Fighter, Computers and Fluids, Vol. 32, (2003), 3-29.

Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD

Beth Plale[1], Dennis Gannon[1], Dan Reed[2], Sara Graves[3], Kelvin Droegemeier[4], Bob Wilhelmson[5], and Mohan Ramamurthy

[1] Indiana University
[2] University of North Carolina, Chapel Hill
[3] University of Alabama Huntsville
[4] Oklahoma University
[5] University of Illinois Urbana Champaign
[6] UCAR, Unidata

Abstract. LEAD is a large-scale effort to build a service-oriented infrastructure that allows atmospheric science researchers to dynamically and adaptively respond to weather patterns to produce better-than-real time predictions of tornadoes and other "mesoscale" weather events. In this paper we discuss an architectural framework that is forming our thinking about adaptability and give early solutions in workflow and monitoring. [7]

1 Introduction

LEAD is a large-scale effort to build infrastructure that allows atmospheric science researchers to dynamically and adaptively respond to weather patterns to produce better-than-real time predictions of tornadoes and other "mesoscale" weather events. This is accomplished by middleware that facilitates adaptive utilization of distributed resources, sensors and workflows, driven by an adaptive event architecture. LEAD is being constructed as a service-oriented architecture. As such, component functionality is encapsulated into individual web services that have well defined interfaces. These services represent both the atomic application tasks as well as the resource and instrument monitoring agents that drive the workflow. The project is broad, with significant effort expended on important efforts such as education and outreach.

The meteorology goal of the project is improved prediction of mesoscale weather phenomena; that is, regional scale weather phenomena such as tornadoes, severe storms and flash floods. The June 1990 outbreak that spawned

[7] LEAD is funded by the National Science Foundation under the following Cooperative Agreements: ATM-0331594 (Oklahoma), ATM-0331591 (Colorado State), ATM-0331574 (Millersville), ATM-0331480 (Indiana), ATM-0331579 (Alabama in Huntsville), ATM03-31586 (Howard), ATM-0331587 (UCAR), and ATM-0331578 (Illinois at Urbana-Champaign). CASA is funded by the NSF under Cooperative Agreement ECE-0313747 to the University of Massachusetts at Amherst

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 624–631, 2005.
© Springer-Verlag Berlin Heidelberg 2005

64 tornadoes across the Midwest and Hurricane Ivan that touched land at Pensacola, FL September 2004 are two examples of mesoscale weather phenomena at the larger end. Improved forecasting of these storms will enable more targeted warnings with longer advance times. Forecasts today are static, that is, generated on a regular schedule, independent of current weather conditions. But important technology and science factors are converging to make significant advancement in forecasting possible. Fast algorithms exists to dynamically detect mesoscale phenomena in data streaming from the Weather Surveillance Radar - 1988 Doppler (WSR-88D) [6]. In a sister project to LEAD, the CASA project [4] is developing small scale regional Doppler radars (*i.e.* NetRad radars) of a size that can be mounted on cell phone towers. These small scale radars have a 30 km radius and sense at a far higher resolution and frequency than the WSR-88D Doppler radar. Finally, large-scale computational grids, such as Teragrid [1], are sufficiently mature in their support infrastructure to where harnessing large numbers of compute resources on demand for application processing is now feasible.

The unique challenge in LEAD is to construct a distributed service framework that is highly responsive to real-time events. Key events are derived from current weather conditions. Specifically, the service framework must be able to respond to weather conditions by directing and allocating resources to collect more information and generate forecasts. Events also occur as execution or runtime phenomena: problems in ingesting data, in network failure, in the resource availability, and in inadequate model progress for instance.

We view real time responsiveness as one enforced by an adaptive infrastructure that mirrors the forecast control flow, while remains largely hidden from user view. While a workflow executes services in the generation of a forecast, the adaptive system is largely hidden, monitoring the behavior of the system, the application, and the external environment. Through generating events that can be understood in the larger context of global behavior, the adaptive system interacts with the workflow system to enact appropriate responses to events. In this paper we introduce a conceptual framework for an adaptive system based on the notion of the *software bus*, a network bus style component that unifies vastly heterogeneous detection components. The actual implementation of the communication medium is still being defined. We also discuss early success in workflow and monitoring.

2 Motivation

Mesoscale meteorology forecasting is both computationally intense and data intense. The following example gives a computer science view of the kind of forecasting that meteorology researchers hope to carry out in the near future.

At 0600hr a research meteorologist kicks off a 2km resolution, 12 hour forecast over the state of Oklahoma. The run is an *ensemble* run in that it consists of 20 copies of the model each with the physics tweaked slightly differently. The runs complete by 0800hr. The final output of the run is 20 binary files containing

temperature, wind, and microphysics. The files undergo statistical analysis to uncover *regions of uncertainty* corresponding to regions across the state where there are high levels of disagreement across the ensemble versions. Such regions may occur when there is too little data available of the region.

To reduce the level of uncertainty, meteorologists want to selectively gather more information about the identified regions. An appropriate outcome of the statistical analysis, then, would be to focus the small NetRad radars in the regions of uncertainty and have them gather additional data. Suppose the radars collect data from 0900hr to 1100hr. The collected data is then converted and assimilated into the 3D input data grid (third and fourth boxes in Figure 1.) Another forecast is kicked off at 1100hr. This time the forecast is a 6 hour forecast, because the forecast need only cover the late afternoon hours when the majority of severe mesoscale weather occurs. The second run finishes at 1300hr. The ensemble results are analyzed again. This time the levels of uncertainty have been reduced so as to give the meteorologist a sufficiently high degree of trust in the forecast.

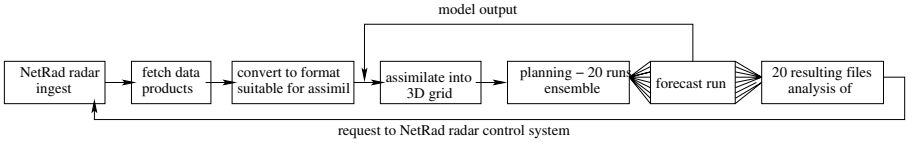


Fig. 1. Control flow for the 20-ensemble run example

This example exposes a number of challenges in provisioning for runs like these. In the flow graph of the example given in Figure 1, for run 1, data products used in the forecast are fetched from where they reside. The ADAS data assimilation system (labeled "assimilate into 3D grid") assumes data products are located in a named directory on the local file system. A forecast might use a dozen or so different data products drawn from Doppler radar, CASA radar, satellite, buoys, balloons, etc. The converted products are assimilated into a single 3D grid. The 3D grid is passed to an ensemble planner who knows the whereabouts of the 20 forecast models with their tweaked physics.

The forecasts are scheduled on computationally intense resources. A single 27 km resolution, 84 hour forecast over the US (a CONUS forecast), consumes 60 processors (30 nodes) of a dual processor Pentium IV Xeon 2.0 GHz cluster at Oklahoma University to run, and takes 6 hours to complete. A regional forecast running an ensemble will presumably require more resources, but due to its smaller area, the scale-up is expected to be less than a factor of 20.

It is important to point out that subsequent loops through the graph in this example are triggered by the outcome of the analysis results of the prior iteration. More pointedly, a second loop iteration is initiated based on meteorological analysis, it is this component that is the source of a directive to the local radars in the areas where uncertainty exists.

3 Adaptive System

Adaptive handling in LEAD is driven by three broad requirements as follows:

- **Highly heterogeneous and numerous sources of adaptive events** – events that have the potential to trigger adaptive behavior occur at the environment level, application level, service level, and at the hardware level,
- **Numerous simultaneous forecasts over limited resources** – 100 or more users may be running forecast models at any one time and concurrently request LEAD resources, and
- **Expecting the unexpected** – workflow can never anticipate every appropriate orchestration response.

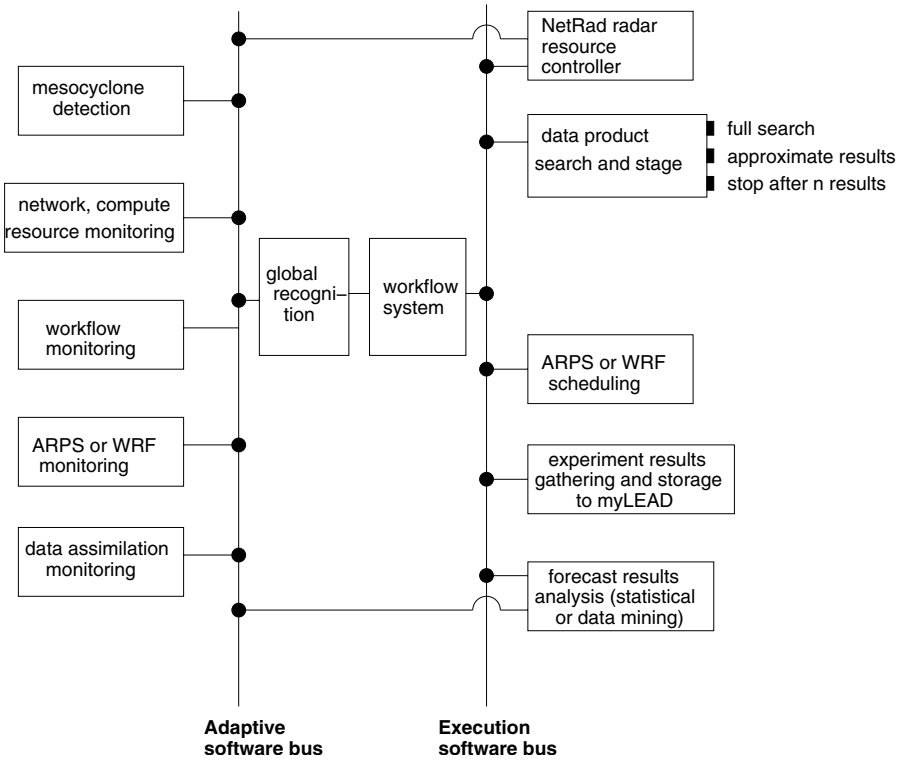


Fig. 2. Active and adaptive software busses. The latter carries events that effect adaptation in the system

Highly heterogeneous and numerous sources of adaptive events. Events that cause adaptive behavior can occur at any level in the system: in the environment when a weather condition arises, in a forecast model analysis that results in directives to a regional radar, at the service layer in response to inef-

ficiencies in an ongoing workflow execution, and at the hardware layer in terms of computational and network loads.

For the system to be responsive to simultaneously occurring high priority events, it employs services to detect, sense, and monitor the environment. These detection services are distributed and gather specialized information close to the source. The mesocyclone detection service (UAH-MDA [6]) might sit on a radar data stream (Level II data) at one of the test bed sites. An instance might also be deployed to examine the 3D assimilated data set generated by a particular experiment. Monitoring detection components can monitor network and compute resources, forecasting components, data assimilation, or the workflow itself.

To achieve optimal adaptation solutions, entities that generate adaptive events must have a mechanism to push those events to a place where they can be considered and acted upon. Heterogeneous components generating reactive events, necessitate an entity that can arbitrate among the competing needs. We can think about the adaptive functionality as a separate system. This is depicted in Figure 2. Where the forecast components to the right are controlled by a workflow system, the adaptive components publish events onto an *adaptive software bus*.

Localized detection and reaction such as might be done by each active component is insufficient because it precludes detection and response to global behavior. For instance, suppose the forecast analysis of Figure 1 identifies two areas of uncertainty over Oklahoma that it wants to resolve, directing the radars to that region for the next 2 hours. But competing interests may have higher priority. Mesocyclone detection may have detected a weather phenomena elsewhere that needs to be investigated with urgency. For instance, the compute resource monitor may note that a particular forecast run is not achieving the level of performance that it needs. But initiating a request for additional compute resources must be balanced against the needs of other users and dynamically occurring events.

Simultaneous forecasts over limited resources. Given the projected scale of LEAD, it is not inconceivable to have 100 or more users simultaneously running forecast models that concurrently request resources. Some of these models may be investigations of a serious weather phenomena. Others might be users tinkering with a model. How does the workflow engine mediate access to the distributed resources? In a data driven setting, workflow must arbitrate access to resources with recognition of urgency and importance.

Expecting the unexpected. A workflow engine can never anticipate every appropriate orchestration response, but in the absence of a specific plan, the workflow may be able to turn to high-level goals that the individual service-level components can help enforce. For instance, one high-level goal may be to *Minimize end-to-end latency*, that is, minimize the total time from when an mesoscale event occurs to when a forecast is available. *Improve accuracy of prediction* is another. If the adaptive capability of the overall system makes it easier to incorporate new data into a forecast, for instance, and that new data

improves the accuracy of the prediction, then the reactive system has contributed to a more accurate forecast. Finally, a suitable goal may be to maximize *Most recent weather data*. If the forecast system requires an infusion of the most recent data obtainable, the reactive system must minimize the time to respond to this need.

Service level components can contribute to enforcing high level goals by implementing and exposing different levels of service. The data search and stage service for instance could implement three levels of response: *full search* where all data resources are searched for relevant products, *approximate results* in which case a fast search returns representative results, or *first results* where the search is conducted at the first location only. Faster searches may result in older or smaller result sets and consequently less accurate forecasts, but in the face of a more urgent need, or the need for a forecast completed as soon as possible, this level of service may be sufficient. These are depicted in Figure 2 as service interfaces to the service "data product search and stage."

4 Workflow in a Reactive System

Basic workflow enactment. Traditional workflow systems operate by executing a fixed schedule of tasks, based on a directed acyclic graph of dependencies. When a task completes, it enables another set of tasks to start operating. Our basic system operates in this manner but with a more flexible architecture than most. A workflow is described in terms of a script described in the industry standard "Business Processing Execution Language" (BPEL) language. A workflow is nothing more than a template for execution of a set of task given a set of parameter bindings. In our case, these parameter bindings correspond to the location, protocol and content of a particular input stream or data file for the workflow template. BPEL allows us to define the workflow in terms of a sequence of well-planed tasks. However, it also allows us to define the task in response to anticipated trigger events.

The life cycle of a simple LEAD BPEL workflow is simple. It starts by executing any requests to services that are available. For example, it may request a resolver service to locate the current data streams of current weather conditions. A request to a service is based on a document-request oriented mode that is unique to our approach. Rather than using the "please-do-this-while-I-wait-for-your-conclusion" remote procedure call (RPC) mechanism, we use the more modern "literal document request" model. The workflow sends a request to an application service. For example, "Please execute the WRF simulation code with these parameters". Rather than wait for a response from the WRF engine giving the results of the computation, the workflow only receives a "request acknowledged" response from the WRF engine. However, to proceed to the next task, the workflow may need to wait for the results.

Application services are programmed to publish status notifications by means of the WS-Eventing web service "publish/subscribe" proposed standard. When the WRF application service completes execution (or fails in its task) it publishes

a "notification" of the result. These notifications are small xml documents that say things like "simulation complete. "

Model for a truly adaptive workflow system. The scenario as described above is inadequate for addressing the adaptive needs of the system. It fails to respond to global conditions and does not yet have the benefit of low level events about the resource performance. The most fundamental limitation of the workflow architecture is that it can only execute workflows that represent programmed responses to anticipated weather scenarios. This is ongoing work and includes considering a balance of the intelligence of the workflow with the intelligence of each component as discussed earlier. The notification system that drives LEAD workflow is a core component of the LEAD Grid middleware. Any client application or service may listen for notifications, act on them and publish new notifications. We can think of the instrument channel as the event bus that is running in parallel with the workflow service notification bus.

5 Dynamic Detection of System-Level Behavior

The monitoring infrastructure is based on Autopilot [7], a toolkit for real-time application and resource monitoring and steering. Autopilot sensor services running on the resources collect performance information at periodic intervals. At each stage the workflow engine publishes events about the progress of the workflow - e.g: Workflow Started, Task A started, Task A ended, etc. The global Control/Monitor service (named "global condition recognition" in Figure 2) acts as a sensor client receiving real-time performance information. The Control/Monitor service also subscribes to events to monitor the progress of the workflow allowing for correlation between collected performance information and the progress of the workflow. It may act as just a simple monitor writing collected data to a file or a more sophisticated scheduling/steering service orchestrating the workflow. The control service may be an actuator client steering the workflow based on the performance monitoring and adaptation or fault monitoring and recovery.

Codes such as WRF instrumented to capture dynamic performance data from hardware performance counter using SvPablo [2] can be streamed to send information through the Autopilot sensor information. The Control/Monitor service may be able to additionally use information from other sensors [8], performance modeling tools to enable dynamic steering of the workflow [5].

6 Dynamic Detection of Mesoscale Phenomena

As part of a larger suite of algorithms to detect mesoscale weather conditions, University of Alabama Huntsville developed the UAH Mesocyclone Detection Algorithm [6] referred to earlier. Mesocyclones contain a velocity signature known as a Rankine Vortex [3], representing incoming and outgoing radial velocity couplets in the WSR-88D data. UAH-MDA identifies shear segments based on this

signature and applies fast classification techniques to isolate mesocyclones in large data sets. The identification accuracy compares favorably to existing MDA algorithms.

7 Conclusion

Adaptability is a key goal to LEAD success. The science goals will not be achievable without an underlying software infrastructure that manages the unexpected events arising in the system with the same attention to detail that it gives to executing a forecast. It is the synergy between the 'active' and 'adaptive' systems that will bring about the flexibility in the system that is critical to advancing meteorology research.

Acknowledgments

The first author would like to thank Jerry Brotzge, Director of NetRad Operations, University of Oklahoma for extensive discussion in identifying interoperability between LEAD and CASA.

References

1. Teragrid. http://www.teragrid.org, 2005.
2. Luiz DeRose, Ying Zhang, and Daniel A. Reed. Svpablo: A multi-language performance analysis system. In *10th International Conference on Computer Performance Evaluation - Modeling Techniques and Tools - Performance Tools'98*, pages 352–355, 1998.
3. R. J. Donaldson. Vortex signature recognition by a doppler radar. *Journal Applied Meteorology*, 9:661–670, 1970.
4. K. Droegemeier J. Kurose D. McLaughlin B. Philips M. Preston S. Sekelsky J. Brotzge, V. Chandresakar. Distributed collaborative adaptive sensing for hazardous weather detection, tracking, and predicting. In *Computational Science - ICCS 2004: 4th International Conference*. June.
5. K. Kennedy, M. Mazina, J. Mellor-Crummey, K. Cooper, L. Torczon, F. Berman, A. Chien, Holly Dail, O. Sievert, D. Angulo, I. Foster, D.Gannon, L. Johnsson, C. Kesselman, J. Dongarra, S. Vadhiyar, R. Wolski, R. Aydt, and D. Reed. Toward a framework for preparing and executing adaptive grid programs. In *Proceedings of the International Parallel and Distributed Processing Symposium Workshop (IPDPS NGS)*. IEEE Computer Society Press, April 2002.
6. Xiang Li, Rahul Ramachandran, John Rushing, and Sara Graves. Mining nexrad radar data: An investigative study. In *American Meteorology Society annual meeting*, 2004.
7. Randy Ribler, Jeffrey Vetter, Huseyin Simitci, and Daniel Reed. Autopilot: Adaptive control of distributed applications. In *7th IEEE Symposium on High- Performance Distributed Computing*, July 1998.
8. Rich Wolski. Dynamically forecasting network performance using the network weather service. *Journal of Cluster Computing*, 1:119–132, January 1998.

Towards a Dynamic Data Driven Application System for Wildfire Simulation

Jan Mandel[1], Lynn S. Bennethum[1], Mingshi Chen[1], Janice L. Coen[2], Craig C. Douglas[3], Leopoldo P. Franca[1], Craig J. Johns[1], Minjeong Kim[1], Andrew V. Knyazev[1], Robert Kremens[4], Vaibhav Kulkarni[1], Guan Qin[5], Anthony Vodacek[4], Jianjia Wu[5], Wei Zhao[5], and Adam Zornes[3]

[1] University of Colorado Denver, Denver, CO 80217-3364, USA
[2] National Center for Atmospheric Research, Boulder, CO 80307-3000, USA
[3] University of Kentucky, Lexington, KY 40506-0045, USA
[4] Rochester Institute of Technology, Rochester, NY 14623-5603, USA
[5] Texas A&M University, College Station, TX 77843-1112, USA

Abstract. We report on an ongoing effort to build a Dynamic Data Driven Application System (DDDAS) for short-range forecast of wildfire behavior from real-time weather data, images, and sensor streams. The system should change the forecast when new data is received. The basic approach is to encapsulate the model code and use an ensemble Kalman filter in time-space. Several variants of the ensemble Kalman filter are presented, for out-of-sequence data assimilation, hidden model states, and highly nonlinear problems. Parallel implementation and web-based visualization are also discussed.

1 Introduction

We report on an ongoing effort to build a Dynamic Data Driven Application System (DDDAS) for short-range forecast of wildfire behavior with models steered by real-time weather data, images, and sensor streams. The forecast should be based on all available data, such as fuel maps, airborne images, internet weather data, field sensor data stream, and raw weather station streams. The forecast should change quickly when new data arrives. Delayed data should be taken into consideration as well and the system should provide measurement steering. The complete system should provide animated visualization and user steering over the internet. In this paper, we are mainly focusing on new developments, perspectives, and practical issues that have emerged since the previous paper [1], where additional details and further references can be found.

The core of the system is a coupled weather-fire model [2], developed by adding a semi-empirical fire model, based on Rothermel fire spread formulas [3], to the Clark-Hall mesoscale atmosphere model [4]. The data needs to be assimilated into the coupled model. However, this software is a legacy FORTRAN code, which is very hard to modify. Instead, we have encapsulated the code and we use it in an Ensemble Kalman Filter (EnKF) scheme [5]. Steering of data

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 632–639, 2005.
© Springer-Verlag Berlin Heidelberg 2005

acquisition can be achieved from the Kalman filter formulas, which can be used to indicate how the variance would be reduced by measurements of different quantities. Assimilation of out-of-order data will be achieved by maintaining the composite time-space state of the simulation over a number of time steps and applying Kalman filter algorithms to the composite state rather than to the state at a single analysis time.

The weather feedback is essential to fire behavior: it is well known that fire creates its own weather. Related work in the literature that combines weather and fire modeling includes FIRETEC [6], which uses a much more detailed (and computationally demanding) physics and chemistry based model, and an implementation of the same fire model as here within the Weather Research and Forecasting (WRF) Model [7]. Practical field tools, which require the input of weather information, such as the wind, include FARSITE [8].

2 Building a Foundation for DDDAS

Current tools used in the field include calculator-based algorithms that predict point-based fire spread from a wind speed, terrain slope, and fuel type as well as two-dimensional PC-based extensions of these formulas [8]. Building awareness of more advanced capabilities that currently exist and of work in progress toward sophisticated systems based on DDDAS is a part of this project.

Thus, we undertook to demonstrate first that atmosphere-fire models several orders of magnitude more complex than current operational tools can be run in real time even on single processor machines. A suite of simulations in which 3–6 nested domains were used to telescope from a 10-km regional-scale weather model to 1.1 km - 41 m horizontal grid spacing respectively were compared, as well as simulations in which the feedback from the fire to the atmosphere was prevented. These demonstrated that this fire-atmosphere feedback was necessary to capture the evolution of the fire shape but that even the relatively coarse grid resolutions could capture the overall growth and progression of the fire. Such simulations with a few hundred meter horizontal grid size can easily be done on a single processor, i.e. such as a PC at a command center, at faster than real time.

Then, in the first application of its kind [9], we have demonstrated that this model could be applied routinely to real fires in Colorado during summer 2004. Upon the operator learning the location of a fire start, the model was configured to establish 3 nested modeling domains over the ignition point. Input data included a locally-run 48-hr regional weather model forecast (MM5), information about the fuel types available for fires across Colorado, regionwide terrain elevation, and fuel moisture noted in incident reports and diagnosed from weather station data. The predicted fire behavior and growth throughout the remainder of the 48-hr forecast was then modeled. Although many refinements can be made in this approach, it is the foundation upon which DDDAS techniques can be built.

3 Software Structure

The DDDAS is built from three main components interacting only through specified interfaces, so the components are independent and any of them can be exchanged with ease. There are multiple versions of each component.

1. The *model* component encapsulates all interaction with a specific model, such as advancing a single simulation to a specified time (Sec. 4), random perturbation of the simulation state, and data acquisition and comparing the data to the model state, i.e., the observation function (Sec. 7, 8). The model component will also interact with the visualization and model steering interface in future (Sec. 10).
2. The *EnKF* component contains an implementation of the filter algorithms in a model independent form (Sec. 5).
3. The *dynamic data assimilation driver* component [1] calls the model and the filter components. Its main function to to coordinate the advancement in time of system states and to present the EnKF with the task of updating an ensemble of state vectors. For this purpose, the driver combines states at multiple times into a single composite state and decides which time snapshots to keep and which to recreate on demand, and controls the advancing of all simulations in time. This idea is related to out-of-sequence particle filters in target tracking applications [10, 11], but similar ideas seem not to have been investigated in the context of meteorology and the much more efficient EnKF. The driver also handles any additional terms (Sec. 5, 6) by calls to model specific functions in the model component.

4 Fire Model

In the current fire model, the burning region in each fuel cell is represented by a quadrilateral defined by four points, called tracers. This submesh representation allows the cell to ignite gradually. The shape of the fireline can be quite complex. However, the EnKF assumes that there is a state u which can be meaningfully increased or decreased, and the tracer information is not such a quantity; for example, the fireline will need to be adjusted over several cells. Therefore, work is in progress to represent the location of the fire as the set of all points x such that $g(x) > 0$, where g is a level function from a finite element space. Level functions have been successfully used in PDE models with free boundary [12]. This work is also the foundation of eventually replacing the empirical fire model with one based on differential equations.

We have developed a simple standalone fire model, based on a finite difference discretization of the reaction-convection-diffusion equation [1]. This model, while admittedly very simple and not quite physical, does produce a reasonable fire behavior with an advancing fire front. The purpose of the simple fire model is to test the performance of ensemble filters on equations with the type of nonlinear behavior that can be expected in a fire, and to serve as the background for

the development of further fire models, with more realistic features such as, e.g., multiple species of fuel and modeling of the reaction intensity as a separate variable from the temperature. Such improved PDE based models will be calibrated against the existing empirically calibrated model, and they will be implemented using parallel Finite Elements and coupled with the weather model.

5 Ensemble Kalman Filter

In the EnKF approach, the model state is a probability distribution, represented by an ensemble of simulations, and the data probability distribution is represented by the measurement values and the associated error estimates. The model state is then updated using the Bayes theorem assuming that the probability distributions are approximately normal, which makes the filter much more efficient.

Consider a simulation state vector u, a measurement vector y, and an observation function h. The value $h(u)$ is what the measurements should be if the simulation and the measurements were accurate. Of course in reality always $h(u) \neq y$, so one looks for u^a, called the analysis, so that magnitudes of the residual $y - h(u^a)$ and of the change in the model state $u - u^a$ are in some sense proportional to the uncertainty in the measurements y and in the simulation state u, respectively. The uncertainty of the measurements is assumed to be known. The uncertainty of the simulation u is estimated by running an ensemble of simulations from initial conditions with random perturbations and observing how far apart they spread. Under the assumption that the probability distributions are normal, the EnKF formulas [5] follow from the Bayes theorem, which coincides in this case with the least squares estimates.

Our first EnKF implementation follows [13]. Further improvements in the EnKF sampling strategies and implementations of the reduced rank analysis scheme based on eigenvalue and singular value techniques have been gaining popularity recently [5, 14]. However, with a few exceptions, e.g., [15], modern efficient eigenvalue and singular value solvers are apparently not well known in the data assimilation community. We are examining the use of the eigensolver LOBPCG [16] and its modifications for low-rank approximation of covariance matrices in the ensemble filter and to reduce the ensemble size by an improved generation of the initial ensemble. Modifications of the EnKF for distributions that are not even approximately normal were presented, e.g., in [17]. The modular structure of our software system allows an easy replacement of the ensemble filter and experiments with various filters, without changing the other software components.

The EnKF is known to work well in oceanography and meteorology; however, its application even to the simple fire model (Sec. 4) fails. The reason is that the simulations in the ensemble become spatially rough and randomly exceed the ignition temperature, then soon the whole ensemble is on fire everywhere. Therefore, we have developed a special filter [18], which controls the value of the spatial gradient of the solution by adding to the least squares function to be

minimized a quadratic form of the difference between the gradient of the solution and the gradient of the mean of the ensemble members prior to the update. This filter was observed to result in low residuals of the analysis ensemble with no spurious fires.

6 Hidden Model Updating

The EnKF methodology assumes that the complete state of the model is modified by the filter, and any modified state is valid. Unfortunately, there are various dependencies between the state variables and constraints on which states are meaningful. For example, the weather model consists of several loosely coupled nested layers with complicated dependencies between them [4]. Just describing these dependencies and constraints outside of the model code would mean bringing much of the physics to the interface and duplicating a significant part of the code. Often, the constraints implemented in the model cannot be described even in principle because the consistency between variables in the model may be enforced only over time by techniques such as nudging [19]. For example, instead of imposing the constraint $u = v$, the evolution equation $\frac{\partial u}{\partial t} = F(u, \ldots)$ for u may be modified to read

$$\frac{\partial u}{\partial t} = F(u, \ldots) - c(u - v), \quad c > 0. \tag{1}$$

Difficulties with the validity of updated states are not widely discussed in the literature; [20] is a notable exception. The common solution seems to be to update some of the variables and simply hope for the best, that the change will propagate to all variables properly after some time.

In addition, it is not desirable to update all variables because there are too many of them. The part of the system state that is not being updated is hidden from the Kalman filter, contrary to its mathematical derivation. We propose to update only a small part of the state and to improve the chance of the hidden variables to settle to their natural values by a combination of nudging and the EnKF. We will compute the new value of the state vector u in each simulation by an ensemble Kalman algorithm, then restart each simulation from an earlier time and impose the new value as v in (1).

7 Assimilation of Airborne Images and Sensor Streams

Airborne near and thermal infrared images can be processed to show the presence of fire quite reliably [21, 22]. The challenges in assimilation of images include the large number of pixels, matching geographical coordinates of the pixels to the coordinates in the model, and recognizing that the errors are correlated because different cameras may be used to image the scene. The ongoing development of algorithms for extracting fire temperature and predicting the direction of fire propagation will provide further data for comparison to the model output [23,

24]. Image processing also allows the transmission of only the pertinent pixels containing a fire signal, greatly reducing bandwidth requirements, while taking into account that false positive fire detections are more likely than false negatives.

Ground sensor data streams that can provide direct fire measurement are now available, but they also must be appropriately matched to the model time step [25]. Techniques used in this area include averaged time derivatives or Fourier coefficients in time.

8 Weather Data Input

Assimilation of weather data requires creating an observation function h such that for a weather model state u, $h(u)$ are the same physical variables and at the same points in space as the available data y. This process involves the transformation of physical variables and units and spatial interpolation. The separation of the observation function into linear and nonlinear terms should be also given, as well as error bounds for the weather data. The error bounds are known from instrument accuracy or from the statistics of the ensembles used for the forecasts.

These physical variables include temperature, pressure, wind speed and direction, and atmospheric humidity. They come from weather data sources that range from three-dimensional weather model analyses and forecasts to point-based weather station data. These are delivered in near real time via the NOAA-PORT satellite broadcast.

9 Parallel Implementation

The code under development will run using coarse grain parallelism. A master controls a very large number of clients that have two tasks: (1) run a weather and wildfire simulation a short period of time, and (2) perform the EnKF update with the weather and wildfire state from (1). The ensemble filter update only consists of matrix-matrix multiplications (dense and sparse). Parallel iterative equation solvers and eigensolvers are also used with distributed data on the clients.

As we incorporate the WRF weather model [7] into our code, we will use its highly tuned parallel implementation. This will allow the weather simulation to execute on a group of nodes. The current wildfire model is strictly serial. Parallelization of the wildfire model consists of replacing the current model with a new, parallel computer friendly PDE based model.

10 Web Based Interface and Visualization

We currently dump physical fields into files out for visualization and/or analysis during the simulation at about 30 second intervals of model time. The 3D fields

are the wind velocity components, buoyancy, pressure perturbation, cloud microphysics fields (water vapor, cloud water, rain, etc.), and smoke concentration. The 2-D fields are the fire heat and water vapor fluxes at each fuel cell and the fuel remaining. These fields can be visualized, e.g., with IDL or Matlab.

In future, we will visualize simulation results on a web site using Java applets. The software structure will support real time visualization with the data streamed to the client, near real-time visualization, where the client periodically retrieves the output of an on-going simulation and examines the process of the spreading fire, as well as offline playback. The same software structure will support clients from powerful workstations, which can provide full 3D visualization for offline analysis, to PDAs and cell phones, which may be limited to 2D animated maps or only periodically refreshed 2D static maps. In any case, the user will be able to pause, rewind, forward, zoom in and out, switch angles, and interact with the visualization to input data.

Separate visible layers will present information such as maps, fuel, wind, fire location and intensity, and probability of burning (from the ensemble forecast). The user will be able to turn the layers on and off, change their order, or modify their transparency. Also, transparency may be used to visualize the probability of the fire, while the color can represent fire intensity.

Acknowledgement

This research has been supported by the National Science Foundation under grants ACI-0325314, 0324989, 0324988, 0324876, and 0324910.

References

1. Mandel, J., Chen, M., Franca, L.P., Johns, C., Puhalskii, A., Coen, J.L., Douglas, C.C., Kremens, R., Vodacek, A., Zhao, W.: A note on dynamic data driven wildfire modeling. In Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J., eds.: Computational Science - ICCS 2004. Volume 3038 of Lecture Notes in Computer Science. Springer (2004) 725–731
2. Clark, T.L., Coen, J., Latham, D.: Description of a coupled atmosphere-fire model. Intl. J. Wildland Fire **13** (2004) 49–64
3. Rothermel, R.C.: A mathematical model for predicting fire spread in wildland fires. USDA Forest Service Research Paper INT-115 (1972)
4. Clark, T.L., Farley, R.D.: Severe downslope windstorm calculations in two and three spatial dimensions using anelastic interactive grid nesting: A possible mechanism for gustiness. J. of the Atmospheric Sciences **41** (1984) 329–350
5. Evensen, G.: The ensemble Kalman filter: Theoretical formulation and practical implementation. Ocean Dynamics **53** (2003) 343–367
6. Linn, R., Reisner, J., Colman, J., Winterkamp, J.: Studying wildfire behavior using FIRETEC. Int. J. of Wildland Fire **11** (2002) 233–246
7. Patton, E.G., Coen, J.L.: WRF-Fire: A coupled atmosphere-fire module for WRF. In: Preprints of Joint MM5/Weather Research and Forecasting Model Users' Workshop, Boulder, CO, June 22–25. NCAR (2004) 221–223

8. Finney, M.A.: FARSITE: Fire area simulator-model development and evaluation. Res. Pap. RMRS-RP-4, Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. 47 p., http://www.farsite.org (1998)
9. Coen, J.L.: Simulation of the Big Elk Fire using using coupled atmosphere-fire modeling. International J. of Wildland Fire **14** (2005) in print
10. Mallick, M., Kirubarajan, T., Arulampalam, S.: Out-of-sequence measurement processing for tracking ground target using particle filters. In: Aerospace Conference Proceedings, 2002. Volume 4., IEEE (2002) 4–1809–4–1818
11. Orton, M., Marrs, A.: A Bayesian approach to multi-target tracking and data fusion with out-of-sequence measurements. In: IEE International Seminar Target Tracking: Algorithms and Applications. Volume 1., IEE, London, UK (2001) 15/1–15/5
12. Beneš, M.: Mathematical and computational aspects of solidification of pure substances. Acta Mathematica Universitatis Comenianae. New Series **70** (2000) 123–151
13. Burgers, G., van Leeuwen, P.J., Evensen, G.: Analysis scheme in the ensemble Kalman filter. Monthly Weather Review **126** (1998) 1719–1724
14. Evensen, G.: Sampling strategies and square root analysis schemes for the EnKF. Ocean Dynamics (2004) 539–560
15. European Centre for Medium-Range Weather Forecasts: Integrated Forecast System (IFS) documentation (CY28r1). V. The Ensemble Prediction System (2004) http://www.ecmwf.int/research/ifsdocs/CY28r1/Ensemble.
16. Knyazev, A.V.: Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. SIAM J. Sci. Comput. **23** (2001) 517–541 (electronic) Copper Mountain Conference (2000).
17. Bengtsson, T., Snyder, C., Nychka, D.: Toward a nonlinear ensemble filter for high dimensional systems. J. of Geophysical Research - Atmospheres **108(D24)** (2003) Art. No. 8775
18. Johns, C.J., Mandel, J.: A two-stage ensemble Kalman filter for smooth data assimilation. Environmental and Ecological Statistics (2005) submitted
19. Kalnay, E.: Atmospheric Modeling, Data Assimilation and Predictability. Cambridge University Press (2003)
20. Bertino, L., Evensen, G., Wackernagel, H.: Sequential data assimilation techniques in oceanography. International Statistical Review G **71** (2003) 223–241
21. Radke, L.R., Clark, T.L., Coen, J.L., Walther, C., Lockwood, R.N., Riggin, P.J., Brass, J., Higgans, R.: The wildfire experiment (WiFE): Observations with airborne remote sensors. Canadian J. Remote Sensing **26** (2000) 406–417
22. Vodacek, A., Kremens, R.L., Fordham, A.J., VanGorden, S.C., Luisi, D., Schott, J.R.: Remote optical detection of biomass burning using a potassium emission signature. International J. of Remote Sensing **13** (2002) 2721–2726
23. Ononye, A., Vodacek, A., Kremens, R.: Improved fire temperature estimation using constrained spectral unmixing. In: Remote Sensing for Field Users. Am. Soc. Photogram. Remote Sens. (2005) CD–ROM Proc. 10th Biennial USDA Forest Service Remote Sensing Applications Conference. Salt Lake City, UT.
24. Vodacek, A., Ononye, A., Wang, Z., Li, Y.: Automatic estimation of direction of propagation of fire from aerial imagery. In: Remote Sensing for Field Users. Am. Soc. Photogram. Remote Sens. (2005) CD–ROM Proc. 10th Biennial USDA Forest Service Remote Sensing Applications Conference. Salt Lake City, UT.
25. Kremens, R., Faulring, J., Gallagher, A., Seema, A., Vodacek, A.: Autonomous field-deployable wildland fire sensors. International J. of Wildland Fire **12** (2003) 237–244

Multiscale Interpolation, Backward in Time Error Analysis for Data-Driven Contaminant Simulation

Craig C. Douglas[1,2], Yalchin Efendiev[3], Richard Ewing[3], Victor Ginting[3], Raytcho Lazarov[3], Martin J. Cole[4], Greg Jones[4], and Chris R. Johnson[4]

[1] University of Kentucky, Department of Computer Science, 325 McVey Hall, Lexington, KY 40506-0045, USA
{craig.douglas, ceshan0}@uky.edu
[2] Yale University, Department of Computer Science, P.O. Box 208285, New Haven, CT 06520-8285, USA
douglas-craig@cs.yale.edu
[3] Texas A&M University, ISC, College Station, TX, USA
{efendiev, ginting, lazarov}@math.tamu.edu, richard-ewing@tamu.edu
[4] Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA
{gjones, mjc}@sci.utah.edu, crj@cs.utah.edu

Abstract. We describe, devise, and augment dynamic data-driven application simulations (DDDAS). DDDAS offers interesting computational and mathematically unsolved problems. In this paper, we discuss how to update the solution as well as input parameters involved in the simulation based on local measurements. The updates are performed in time. We test our method on various synthetic examples.

1 Introduction

In recent years, immense computing power has become available at the national and international supercomputer centers and local clusters of fast PCs. We also have had a proliferation of data acquisition and generation through the deployment of sophisticated new generations of sensors. The lack of coordination between current computational capacity and sensor technology impairs our ability to effectively utilize the flood of information available. This is a substantial barrier to achieving the potential benefit computational science can deliver to many application areas including contaminant tracking, wildfire modeling, transportation optimization, and many other fields.

Sensors and data generating devices may take many forms including other running computational simulations. The intent of this paper is to address several DDDAS enabling technologies in the context of a specific application area in order to provide techniques and tools to effectively demonstrate the potential

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 640–647, 2005.
© Springer-Verlag Berlin Heidelberg 2005

of dynamic data driven simulations for other areas. Our primary application is contaminant tracking, which in groundwater reservoirs is modeled by strongly coupled systems of convection-reaction-diffusion equations. The solution process of such systems becomes more complicated when modeling remediation and clean-up technologies since they exhibit strong nonlinearities and local behavior. Many of these applications are essentially computer models that solve nonlinear, unsteady, coupled, partial differential equations. All require consistent initial conditions, adequate forcing fields, and boundary conditions to advance the solution in time. Collectively these fields represent the input data necessary to run the models. The input data can originate from observations, e.g., sensor based telemetry, can be internally generated from ensemble type simulations, or can be externally generated (e.g., providing boundary conditions for very high resolution regional models). The skill of these models to adequately represent realistic conditions is intimately tied to the quality, spatial and temporal coverage, and intelligent use of their input data sets. These applications in turn generate large amounts of output data that must be either analyzed or passed on to other more specialized subcomponents.

The update is performed based on the sensor measurements, which streamed from few spatial locations. As data is injected, we propose to update (1) the solution (2) the initial condition. We have also considered the update of the media properties, but this will not be discussed in this paper. Because of the heterogeneities of the porous media, we employ multiscale interpolation technique for updating the solution. This is done in the context of general nonlinear parabolic operators that include subsurface processes. The main idea of this interpolation is that we do not alter the heterogeneities of the random field that drives the contaminant. Rather based on the sensor data we rescale the solution in a manner that it preserves the heterogeneities. This rescaling uses the solution of the local problems. We compare the numerical results for simulations that employ both updating the data at sensor location and the simulations that do not update the locations. The simulation results that do not use the update or use it less frequently produces large errors. This was observed in our numerical studies.

The errors in the simulations will persist if one does not change the input parameters. As new data are obtained from sensors measurements, the initial data needs to be updated. This update reduces the computational errors associated with incorrect initial data and improves the predictions. In this paper, we consider linear subsurface flows involving convection and diffusion. Initial data is sought in a finite dimensional space. Using the first set of measurements, the approximation of the initial data is recovered. As new data are incorporated into the simulator, we update the initial data using an objective function. We note that the formulated problem is ill-posed. Two facts can be attributed to this ill-posedness. First, the data gathered from the sensor measurements always contain some defects that come from factors such as human errors and inherent factory errors of the sensors. Secondly, the number of sensors that can be installed are limited, and in general are much fewer than the finite dimensional space describing the initial data. For the latter, we can regularize the problem by

using the prior information about the initial data. This prior information is the updated initial data. The penalization constants depend on time of update and can be associated with the relative difference between simulated and measured values. Numerical examples are presented in the paper showing the improvement of the predictions as new data is taken into account.

2 Backward Error Analysis and Initial Data Recovery

As new data is injected from sensor measurements, "the initial data" can be updated. Here, by initial data we mean contaminant distribution at some early time. Due to poor knowledge of the initial location of contaminant, this type of errors can be dominant in simulations. We consider a linear transport dominated by convection and diffusion

$$\frac{\partial C}{\partial t} + v \cdot \nabla C - \nabla \cdot (D \nabla C) = 0 \text{ in } \Omega, \tag{1}$$

where by Darcy's Law, we have $v = -k\nabla p$, with the pressure p satisfies

$$-\nabla \cdot (k \nabla p) = 0 \tag{2}$$

with some prescribed boundary conditions and initial condition/data $C(\mathbf{x}, 0) = C^0(\mathbf{x})$. Here the variable $C(\mathbf{x}, t)$ is designated for a contaminant concentration over the porous medium Ω and at time level t, k is the permeability of the porous medium, and D is the diffusion coefficient.

We seek the initial data in a finite dimensional space. The dimension of this space is an important factor in our simulations. The problem becomes more ill-posed if this dimension increases. In general, one can reduce this dimension using multiscale representation of the initial data (note that initial data represents the solution at early times, not necessarily at time zero). As new data are incorporated into the simulator, we update the initial data using an objective function. Before we formulate the objective function, we note that the formulated problem is ill-posed. Two facts can be attributed to this ill-posedness. First, the data gathered from the sensor measurements always contain some defects that come from factors such as human errors and inherent factory errors of the sensors. Secondly, the number of sensors that can be installed are limited, and in general are much fewer than the finite dimensional space describing the initial data. For the latter, we can regularize the problem by using the prior information about the initial data. This prior information is the updated initial data. The penalization constants depend on time of update and can be associated with the relative difference between simulated and measured values.

To formulate the objective function we introduce some notations. Let N_s be the number of sensors installed in various points in the porous medium and $\{\mathbf{x}_j\}_{j=1}^{N_s}$ denote such points. Let N_t be the number of how many times the concentration is measured in time and $\{t_k\}_{k=1}^{N_t}$ denote such time levels. Furthermore,

$\gamma_j(t_k)$ denotes the measured concentration at sensor located in $\mathbf{x}_j$ and at time t_k. We seek initial data in a finite dimensional space spanned by $\tilde{C}_i^0(\mathbf{x})$,

$$\tilde{C}^0(\mathbf{x}) = \sum_{i=1}^{N_c} \alpha_i \tilde{C}_i^0(\mathbf{x}), \tag{3}$$

for some $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_{N_c})$. Furthermore, let $\tilde{C}_i(\mathbf{x}, t)$ be the solution of (1) using an initial condition $\tilde{C}_i^0(\mathbf{x})$. Then by superposition principle, the solution of (1) using $\tilde{C}^0(\mathbf{x})$ in (3) as an initial condition has the following form:

$$\tilde{C}(\mathbf{x}, t) = \sum_{i=1}^{N_c} \alpha_i \tilde{C}_i(\mathbf{x}, t). \tag{4}$$

The objective function which can be formulated in terms of α, quantifies the difference between the measured concentration and the simulated concentration, $\tilde{C}(\mathbf{x}, t)$. In general the number of the sensors are less than the dimension of the space for initial data. Hence, an attempt to minimize an objective function that only contains the difference between measurements and simulations will lead to an ill-posed problem. To regularize the problem, we add a penalty term that contains the prior information related to the initial data, and consider the following objective function

$$F(\alpha) = \sum_{j=1}^{N_s} \left(\sum_{i=1}^{N_c} \alpha_i \tilde{C}_i(\mathbf{x}_j, t) - \gamma_j(t) \right)^2 + \sum_{i=1}^{N_c} \kappa_i \left(\alpha_i - \beta_i \right)^2. \tag{5}$$

Here $\kappa = (\kappa_1, \kappa_2, \cdots, \kappa_{N_c})$ is the penalty coefficients for an *a priori* vector $\beta = (\beta_1, \beta_2, \cdots, \beta_{N_c})$. This prior information will be updated during the simulation to achieve higher accuracy.

Next we present a representative numerical example. More numerical and theoretical studies have been performed and the results will be reported elsewhere. We use $\Omega = [0, 1] \times [0, 1]$. The boundary conditions in the subsurface flow for the pressure equation (2) are given pressure at the inlet and outlet edges (i.e., $x = 0$ and $x = 1$, respectively), and no flow at the bottom and top edges (i.e., $z = 0$ and $z = 1$, respectively). The permeability k is generated with given correlation length $l_x = 0.25$ and $l_z = 0.02$, with a spherical variogram using GSLIB algorithms [2]. For the convection-diffusion equation (1), we set the diffusion coefficient $D = 0.1$ over all domain. We assume zero concentration at the inlet, bottom, and top edges, and a zero diffusion, i.e., $(D\nabla C) \cdot \boldsymbol{n} = 0$, at the outlet edge, with $\boldsymbol{n}$ being the unit normal vector pointing outward on the outlet edge. The initial condition $C^0(x, z)$ is set to be nonzero in the region $(0.2, 0.4) \times (0.2, 0.4)$ and zero elsewhere.

Both pressure and convection-diffusion equations are solved by the finite volume method using rectangular grids. We discretize the domain into 100×100 elements, i.e., 100 elements in each direction. The sensors information are obtained from the concentration solved with the given "true" initial condition. We

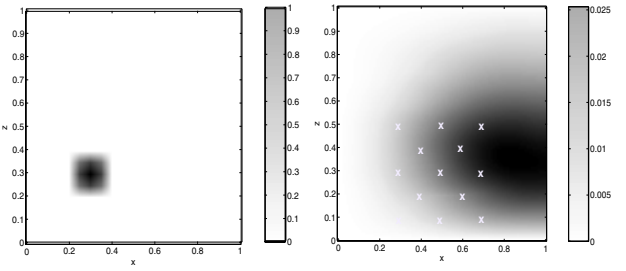


Fig. 1. Left: The initial condition profile - Right: Concentration at t=0.4 ((x) indicates the sensor location)

use time step $\Delta t = 0.01$. For the example presented in this paper we use initial condition shown on the right side of Figure 1.

The data from measurement are taken from multiple set of numerical simulations with the initial condition mentioned earlier. The measurement is assumed to be conducted at time level $t_1 = 0.1$, $t_2 = 0.2$, $t_3 = 0.3$, and $t_4 = 0.4$. A number of sensors are installed at various locations in the porous medium. Figure 1 shows the concentration profile at $t = 0.4$ along with the sensor locations which are denoted by (x) indicator.

We have performed numerical tests by only updating β as well as updating β and κ. In all cases, we observed significant improvement as the initial data is updated. Here we present only the case of both β and κ are updated. In particular, κ is increased during the simulations to reflect the fact that the updated initial condition is a better representation of the true initial condition. In general, one can change κ in various ways. In our examples, we update κ_i by increasing 10 times after each update. Figure 2 shows the updated initial condition with both β and κ updated for the case of larger support. The prior for κ is $\kappa_i^0 = 2 \times 10^{-12}$ for all i, and when updated it is multiplied by ten. The figure shows significant improvement on the predicted initial condition.

As mentioned earlier, sensor measurements contain errors and uncertainties. In our numerical simulations, we can take into account these uncertainties by sampling the sensor data from the known distribution. As a result, one obtains

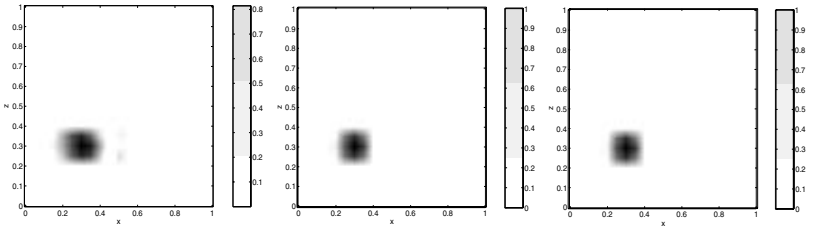


Fig. 2. Updated initial condition: $t = 0.1$ (left), $t = 0.3$ (middle), $t = 0.4$ (right) - Both β and κ are updated

various realizations of the initial data. In our subsequent work [4], we employ the least squares approach in developing Bayesian methods for nonlinear problems. To quantify uncertainties in the measurements and *a priori* knowledge about the initial data, the Markov Chain Monte Carlo method (MCMC) can be used. Because this method is expensive due to rejection of the proposals, we propose an approach that combines the least squares method with Bayesian approaches that will give high acceptance rates.

3 Multiscale Interpolation Techniques

Our goal in this section is to discuss the mapping of the sensor data to the finite dimensional space where the solution is calculated. This procedure is nontrivial in general, because the solution space usually has high dimension, while the sensors are located only at few locations. Our simplified approach presented in this paper consists of passing the sensor data to the simulations and its use for the next time step simulations. Since the sensor data represents the solution only at few coarse locations one has to modify the solution conditioned to this data. This step we call multiscale interpolation which consists of mapping the sensor data to the solution space. At each time step the sensor data is received by the simulator. There are two options to handle the data. We can treat it as hard data or as soft data. The latter means that the data contains some noise and not needed to be imposed exactly. In this paper the first approach, "hard constraint", will be considered. At the beginning of each time step we need to map the sensor data to the solution space. This is performed using DDDAS mapping operator, the main feature of which is not to alter the heterogeneous field.

The proposed mapping for the sensor data is general and applicable to various classes of equations. To demonstrate this we consider general nonlinear parabolic equations

$$\frac{\partial}{\partial t}u_\epsilon = \nabla \cdot (a_\epsilon(x,t,u_\epsilon,\nabla u_\epsilon)) + a_{0,\epsilon}(x,t,u_\epsilon,\nabla u_\epsilon), \text{ in } \Omega \times [0,T], \tag{6}$$

where ϵ indicates the presence of the small scales heterogeneities. This equation includes various physical process that occur in subsurfaces. In the next section numerical examples for particular cases of (6) will be discussed. Assume the domain is divided into the coarse grid such that the sensor points are the nodal points of the coarse grid. Note that we do not require all nodal points to be sensor locations. Further denote by S^h the space of piecewise linear functions on this partition,

$$S_h = \{v_h \in C^0(\overline{\Omega}) : \text{the restriction } v_h \text{ is linear for each triangle } K \in \Pi_h\}.$$

Our objective now is to map the function defined on S^h to the fine grid that represents the heterogeneities. This grid is obtained from *apriori* information about

the field using geostatistical packages. Denote by the operator E the mapping from the coarse dimensional space into the fine grid,

$$E : S^h \to V_\epsilon^h,$$

which is constructed as follows. For each element in $u_h \in S^h$ at a given time t_n we construct space time function $u_{\epsilon,h}(x,t)$ in $K \times [t_n, t_{n+1}]$ such that it satisfies

$$\frac{\partial}{\partial t} u_{\epsilon,h}(x,t) = \nabla \cdot (a_\epsilon(x,t,\eta,\nabla u_{\epsilon,h})) \tag{7}$$

in each coarse element K, where η is the average of u_h. $u_{\epsilon,h}(x,t)$ is calculated by solving (7) on the fine grid, and thus it is a fine scale function. To complete the construction of E we need to set boundary and initial conditions for (7). One can set different boundary and initial conditions and this will give rise to different maps. In our numerical simulations we will take the boundary and initial condition for the local problems to be linear with prescribed nodal values. These values are obtained from the sensor data, if available. If the sensor data is not available at some location we use the values obtained from the simulations. Different local boundary conditions can be also imposed and will be discussed later. Mathematical aspects of this interpolation operator, such as convergence and etc, are described in [5].

Once the solution at time $t = t_n$ is computed its values with sensor data at the sensor locations are compared. After changing the values of the solution we interpolate it to the fine grid and use it for the next time step. At the last step we use multiscale approach which is computationally efficient. In particular, the solution at the next time step is calculated based on

$$\int_\Omega (u_h(x,t_{n+1}) - u_h(x,t_n))v_h dx + \sum_K \int_{t_n}^{t_{n+1}} \int_K ((a_\epsilon(x,t,\eta,\nabla u_{\epsilon,h}),\nabla v_h) + a_{0,\epsilon}(x,t,\eta,\nabla u_{\epsilon,h})v_h)dxdt = \int_{t_n}^{t_{n+1}} \int_\Omega f dxdt. \tag{8}$$

Here Ω refers to the spatial domain and K are the coarse elements. We would like to note that our approach has limitations and it is not applicable when there are large deviations between sensor data and the solution.

We now present a representative numerical example that demonstrates the accuracy and limitations of our proposed method. The detailed numerical studies will be reported elsewhere. The systems we consider are intended to represent cross sections (in $x - z$) of the subsurface. The fine-scale permeability field is generated on 121×121 grid using GSLIB algorithms [2] with an exponential covariance model. We consider $\frac{\partial}{\partial t} u = \nabla \cdot (a_\epsilon(x)\nabla u)$, where the original "true" diffusion coefficient $a_\epsilon(x) = \exp(\alpha_\epsilon(x))$, where $\alpha_\epsilon(x)$ is a realization of the random field with correlation lengths $l_x = 0.3$, $l_z = 0.02$ and variance $\sigma = 0.75$. For the simulation purposes we consider the diffusion coefficients to be the same realization of the random field but with $\sigma = 1.5$. Thus we assume that the heterogeneities have the same nature and the only difference between the true field

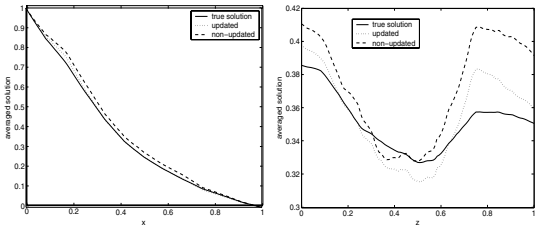


Fig. 3. Comparisons of the average solutions across x and z directions: Solid line designates the true solution, dotted line designates the solution obtained using our simulations with 4 updates, and the dashed line designates the solution that has not been updated

and the one used in the computations is associated with the scaling. Objective of this numerical results is to demonstrate how frequency of updating sensor data in the simulations improves the accuracy of the method. In this example the sensor data is used four times during simulations, i.e., the frequency of updating is 4. In Figure 3 we plot the average of the solutions. Solid line designates the fine scale (true) simulation results, while dotted line represents the results obtained using our methodology with 4 updating. The dashed line represents the simulation results with no updating. As we see from this figure the simulation with data updating performs better compare to that with no updating. The l_2 error between the true solution and the one corresponding with 4 updating is about 6 percent, while l_2 error corresponding with no updating is almost 9 percent. Simulations with more frequent update indicate that the frequent updating improve the accuracy of the predictions and thus important for DDDAS. This is also observed in various numerical examples for both linear and nonlinear equations which we have tested (see [3]).

References

1. J. Bear, *Dynamics of Fluids in Porous Media*, Elsevier, 1972.
2. C. V. Deutsch and A. G. Journel, *GSLIB: Geostatistical software library and user's guide, 2nd edition*, Oxford University Press, New York, 1998.
3. C. C. Douglas, C. Shannon, Y. Efendiev, R. Ewing, V. Ginting, R. Lazarov, M. Cole, G. Jones, C. Johnson, and J. Simpson, *A note on data-driven contaminant simulation*, Lecture Notes in Computer Science, Springer-Verlag, 3038 (2004), pp. 701–708.
4. C. C. Douglas, Y. Efendiev, R. Ewing, V. Ginting, and R. Lazarov, *Bayesian approaches for initial data recovery in dynamic data-driven simulations*, in preparation.
5. Y. Efendiev and A. Pankov. Numerical homogenization of nonlinear random parabolic operators. *SIAM Multiscale Modeling and Simulation*, 2(2):237–268, 2004.
6. C.R. Johnson, S. Parker, and et al. SCIRun: A scientific computing problem solving environment. http://software.sci.utah.edu/scirun.html.

Ensemble–Based Data Assimilation for Atmospheric Chemical Transport Models *

Adrian Sandu[1] Emil M. Constantinescu[1], Wenyuan Liao[1], Gregory R. Carmichael[2], Tianfeng Chai[2], John H. Seinfeld[3], and Dacian Dăescu[4]

[1] Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061
{asandu, emconsta, liao}@cs.vt.edu
[2] Center for Global and Regional Environmental Research, The University of Iowa, Iowa City, 52242-1297
{gcarmich, tchai}@cgrer.uiowa.edu
[3] Department of Chemical Engineering, California Institute of Technology, Pasadena, CA 91125
seinfeld@caltech.edu
[4] Department of Mathematics and Statistics, Portland State University
daescu@pdx.edu

Abstract. The task of providing an optimal analysis of the state of the atmosphere requires the development of dynamic data-driven systems (D^{3}AS) that efficiently integrate the observational data and the models. In this paper we discuss fundamental aspects of nonlinear ensemble data assimilation applied to atmospheric chemical transport models. We formulate autoregressive models for the background errors and show how these models are capable of capturing flow dependent correlations. Total energy singular vectors describe the directions of maximum errors growth and are used to initialize the ensembles. We highlight the challenges encountered in the computation of singular vectors in the presence of stiff chemistry and propose solutions to overcome them. Results for a large scale simulation of air pollution in East Asia illustrate the potential of nonlinear ensemble techniques to assimilate chemical observations.

Keywords: Dynamic data-driven applications and systems (D^{3}AS), data assimilation, background covariance, ensemble Kalman filter, total energy singular vectors, autoregressive processes.

1 Introduction

Our ability to anticipate and manage changes in atmospheric pollutant concentrations relies on an accurate representation of the chemical state of the

* This work was supported by the National Science Foundation through the award NSF ITR AP&IM 0205198 managed by Dr. Frederica Darema.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 648–655, 2005.
© Springer-Verlag Berlin Heidelberg 2005

atmosphere. As our fundamental understanding of atmospheric chemistry advances, novel data assimilation tools are needed to integrate observational data and models together to provide the best, physically consistent estimate of the evolving chemical state of the atmosphere. Data assimilation is vital for meteorological forecasting and has started to be applied in chemical transport modeling [7, 10, 20, 24].

In this paper we focus on the particular challenges that arise in the application of nonlinear ensemble filter data assimilation to atmospheric chemical transport models (CTMs). The distinguishing feature of CTMs is the presence of nonlinear and stiff chemical interactions occurring at characteristic time scales that are typically much shorter than the transport time scales. CTMs propagate the model state forward in time from the "initial" state $x(t_0)$ to the "final" state $x(t_\mathrm{F})$ (1). Perturbations (small errors) evolve according to the tangent linear model (2) and adjoint variables according to the adjoint model (3):

$$x(t_\mathrm{F}) = \mathcal{M}_{t_0 \to t_\mathrm{F}} \ (x(t_0)) \tag{1}$$

$$\delta x(t_\mathrm{F}) = M_{t_0 \to t_\mathrm{F}} \ \delta x(t_0) \tag{2}$$

$$\lambda(t_0) = M^*_{t_\mathrm{F} \to t_0} \lambda(t_\mathrm{F}) \ . \tag{3}$$

Here $\mathcal{M}$, M, and M^* denote the solution operators of the CTM, the tangent linear, and the adjoint models respectively. The error covariance matrix evolves from $P(t_0)$ to $P(t_\mathrm{F})$ according to

$$P(t_\mathrm{F}) = M_{t_0 \to t_\mathrm{F}} \, P(t_0) \, M^*_{t_\mathrm{F} \to t_0} + Q \ , \tag{4}$$

where Q is the covariance of the model errors.

Kalman filter techniques [16] provide a stochastic approach to the data assimilation problem. The filtering theory is described in Jazwinski [15] and the applications to atmospheric modeling in [6, 19]. The computational burden associated to the filtering process has prevented the implementation of the full Kalman filter for large-scale transport-chemistry models. Ensemble Kalman filter techniques [8, 9, 13] may be used to facilitate the practical implementation as shown by van Loon et al. [24].

In the ensemble implementation of the Kalman filter [9] the statistics are represented by the ensemble mean and covariance. These statistics depend strongly on the background (initial) ensemble statistics $x(t_0)$ and $P(t_0)$. Since the probability density of the background state is not known exactly, it needs to be modeled. Previous efforts to develop flow dependent background covariance models are due to Riishojgaard [21], Hamill et al. [11], Houtekamer et. al. [14], and Buehner et. al. [1].

This paper brings the following new elements:

1. The background errors are modeled using autoregressive processes. Such models are computationally inexpensive and capture the error correlations along the flow lines.
2. Total energy singular vectors (TESVs) are calculated for chemically reactive flows. TESVs are the directions of maximum error growth over a finite time interval.

3. The above techniques are used to initialize the ensembles in a large scale data assimilation problem.

The paper is organized as follows. Section 2 presents the background error models and the calculation of TESVs. Section 3 illustrates the use of the tools in a large scale data assimilation test, and Section 4 summarizes the results of this research.

2 Construction of the Initial Ensemble

A good approximation of the background error statistics and a correct initialization of the ensemble are essential for the success of ensemble data assimilation. In this section we consider autoregressive models for background errors and discuss the construction of TESVs. A more detailed discussion can be found in [5] and [18].

2.1 Modeling the Background Errors

The background state x^{B} is represented as the sum of the average state $\overline{x}^{\mathrm{B}}$ plus an error (uncertainty) field δx^{B}, $x^{\mathrm{B}} = \overline{x}^{\mathrm{B}} + \delta x^{\mathrm{B}}$. The error field has zero mean $\langle \delta x^{\mathrm{B}} \rangle = 0$, and background covariance $B = \langle \delta x^{\mathrm{B}} \left(\delta x^{\mathrm{B}}\right)^T \rangle$. Our basic assumption is that the background state errors form a multilateral autoregressive (AR) process [12] of the form

$$\delta x^{\mathrm{B}}_{i,j,k} = \alpha_{\pm 1} \delta x^{\mathrm{B}}_{i\pm 1,j,k} + \beta_{\pm 1} \delta x^{\mathrm{B}}_{i,j\pm 1,k} + \gamma_{\pm 1} \delta x^{\mathrm{B}}_{i,j,k\pm 1} + \sigma_{i,j,k}\, \xi_{i,j,k} \ . \qquad (5)$$

Here (i, j, k) are gridpoint indices on a 3 dimensional (structured) grid. The model (5) captures the correlations among neighboring grid points, with α, β ,γ representing the correlation coefficients in the x, y and z directions respectively. The last term represents the additional uncertainty at each grid point, with $\xi \in \mathcal{N}(0, 1)$ normal random variables and σ local error variances. The motivation behind multilateral AR models is the fact that (5), – with proper coefficients – can be regarded as a finite difference approximation of the advection-diffusion equation.

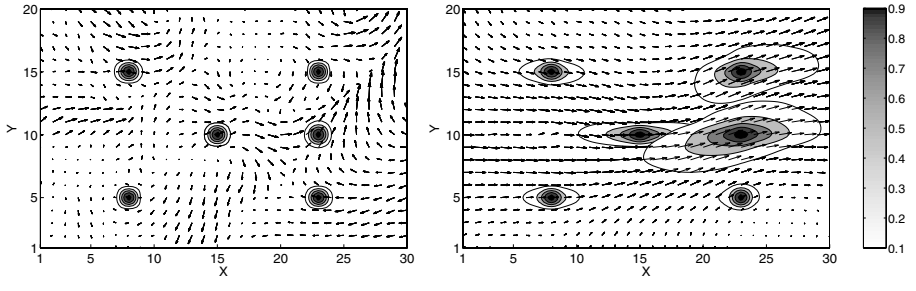
The AR process (5) can be represented compactly as

$$A\, \delta x = \xi \ . \qquad (6)$$

Note that A is a very sparse matrix. The background error covariance matrix is $B = A^{-1} A^{-T}$, and the correlation matrix is $D = \mathrm{diag}(B)^{-1/2} B \mathrm{diag}(B)^{-1/2}$.

Constant correlation coefficients α, β, γ imply fixed spatial directional correlation whereas variable coefficients may be used to capture flow dependent correlations. In this paper we use the scaled wind speeds u, v, and w as correlation coefficients. For example, the correlation coefficients in the x direction are given by $\alpha_{i,j,k} = u_{i,j,k} / \max_{i,j,k}(\sqrt{u^2_{i,j,k} + v^2_{i,j,k} + w^2_{i,j,k}})$. This approach leads to very well conditioned covariance matrices B.

To illustrate the autoregressive models we consider the wind fields over East Asia on 0 GMT, March 1st, 2001, corresponding to the Trace-P field campaign [3]. An autoregressive model (5) of background errors is constructed using flow dependent coefficients (scaled wind velocities). Top views of the spatial correlations of the resulting uncertainty fields are shown in Figure 1 for several gridpoints located on the ground layer (a) and on the top layer (b). The correlations match the shape and magnitude of the wind field. Note that the wind speed near the ground is smaller than at the top and this is reflected by the correlations.



(a) Ground Level (b) Top Level

Fig. 1. Background error correlations for the Trace-P wind fields on March 1, 2001

2.2 Total Energy Singular Vectors

Total energy singular vectors (TESVs) are the directions of the most rapidly growing perturbations over a finite time interval. We measure the magnitude of the perturbations in the concentration fields using L^2 ("energy") norms. The ratio between perturbation energies at the final (t_{F}) and initial time (t_0) offers a measure of error growth:

$$\sigma^2 = \frac{\|\delta x(t_{\mathrm{F}})\|_B^2}{\|\delta x(t_0)\|_A^2} = \frac{\langle \delta x(t_0), M^*_{t_{\mathrm{F}} \to t_0} B M_{t_0 \to t_{\mathrm{F}}} \delta x(t_0) \rangle}{\langle \delta x(t_0), A \delta x(t_0) \rangle} \tag{7}$$

Here A is a positive definite and B a positive semidefinite matrix. In (7) we use the fact that perturbations evolve in time according to the dynamics of the tangent linear model (2). TESVs are defined as the directions of maximal error growth, i.e. the vectors $s_k(t_0)$ that maximize the ratio σ^2 in equation (7). These directions are the solutions of the following generalized eigenvalue problem:

$$M^*_{t_{\mathrm{F}} \to t_0} \, B \, M_{t_0 \to t_{\mathrm{F}}} \, s_k(t_0) = \sigma_k^2 \, A \, s_k(t_0) \tag{8}$$

The left side of (8) involves one integration with the tangent linear model followed by one integration with the adjoint model.

The eigenvalue problem (8) is solved by software packages like ARPACK [17] using Lanczos iterations. The symmetry of the matrix $M^* \, B \, M$ required by

Lanczos imposes to use the discrete adjoint M^* of the tangent linear operator M in (8). The computation of discrete adjoints for stiff systems is a nontrivial task [22]. In addition, computational errors (which can destroy symmetry) have to be small. A considerable loss of symmetry during the stiff transient is observed in practice [18]. This is due to the fact that the initial perturbations are away from the slow (non-stiff) manifold. To correct this we apply the tangent linear model on the initial perturbation for a short time, which is equivalent to "projecting" the initial perturbation onto the slow evolution manifold. In order to preserve operator symmetry, another projection using the adjoint model needs to be performed at the end of the adjoint integration. Consequently the matrix-vector products are computed as $w = \Pi^* M^*_{t_F \to t_0} B M_{t_0 \to t_F} \Pi x$, where Π and Π^* denote the projection operations performed with the tangent linear and the adjoint models respectively.

3 Numerical Results

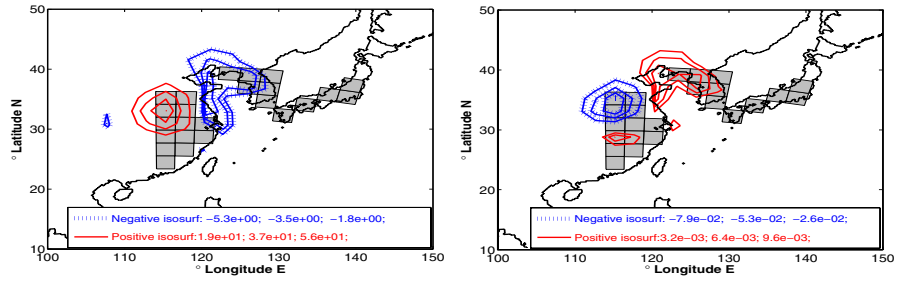
The numerical tests use the state-of-the-art regional atmospheric chemical transport model STEM [3]. The simulation covers a region of 7200 Km × 4800 Km in East Asia and uses a $30 \times 20 \times 18$ computational grid with a horizontal resolution of 240 Km × 240 Km. The chemical mechanism is a variant of SAPRC-99 [4] and accounts for 93 different chemical species. The simulated conditions during March 1$^{\text{st}}$, 2001, correspond to the Trace-P [3]) field experiment.

We consider artificial observations generated every 6 hours by a reference simulation starting at 0 GMT, March 1$^{\text{st}}$, 2001. The observations are ground level ozone (O_3) concentrations at 24 gridpoints over Japan, Korea, and East China. These grid points are referred to as the "target area" (the gray area in Figure 2).

For the calculation of TESVs the final perturbation energy measures the perturbations of (O_3) and nitrogen dioxide (NO_2) in the target area at the final time. The perturbation norm at the initial time accounts for perturbations in all chemical species, scaled by typical concentration values [18]. The O_3 and NO_2 sections of the dominant TESV are shown in Figure 2. We notice that dominant TESV is localized near the target area, and that it is strongly correlated with the adjoint variable corresponding to a similar target function.

The data assimilation process uses an ensemble with 50 members. The ensemble is run for 6 hours in forecast mode, then is analyzed using the artificial observations in the ensemble Kalman framework [9]. The assimilated ensemble is then advanced in time for another 6 hours, then analyzed again, etc. until the end of the 24 hours simulation interval.

Different initial perturbations are considered as follows. The first simulation ("D") uses an uncorrelated background. The initial perturbation is of the form $\delta x^B = 30\%\ x^B \cdot \xi$, where $\xi \in \mathcal{N}(0,1)$ and x^B is the initial concentration vector. The second simulation ("AR") uses a flow dependent AR model for background errors. The initial perturbation is $\delta x^B = A^{-1}\ \left(30\%\, x^B \cdot \xi\right)$, as described in section 2. The third simulation ("AR+SV") adds perturbations along the largest 40

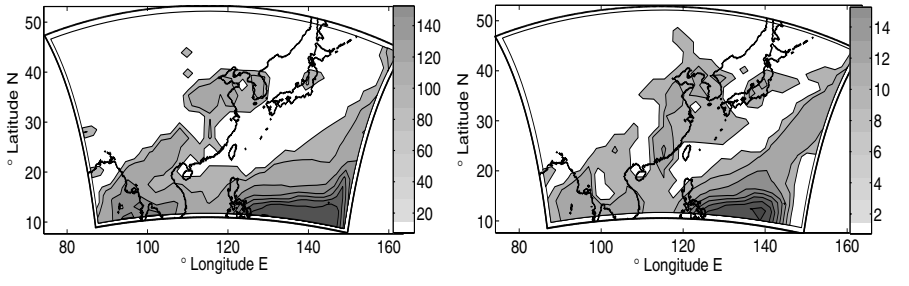


(a) O_3 Section (b) NO_2 Section

Fig. 2. The dominant TESV (for ground level O_3 concentration in the gray area) after 24 hours of evolution

TESVs to the flow dependent perturbations given by the autoregressive model. The TESV perturbations undergo the maximum growth over a 24 hour interval. Reducing uncertainty along these directions impacts the overall accuracy improvements obtained through data assimilation.

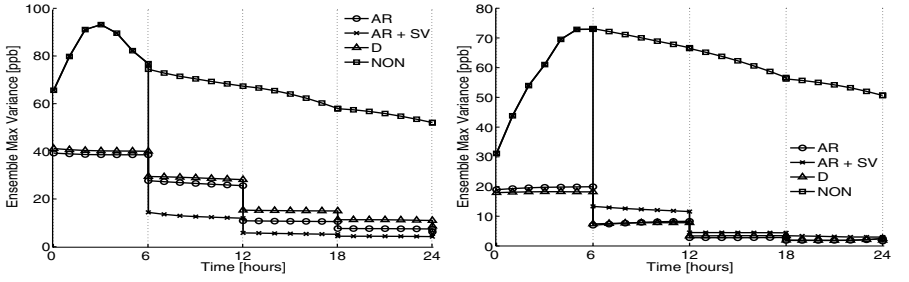
Figure 3 shows the ensemble standard deviation at ground level at the initial and final times using *AR+SV* background perturbations. Data assimilation leads to a large decrease in the ensemble standard deviation after 24 hours.



(a) Initial Time (b) Final Time (w/ Data Assimilation)

Fig. 3. Ensemble standard deviation at ground level with *AR+SV* background perturbations

The 24 hours time evolution of the ensemble O_3 standard deviation over the entire domain is shown in Figure 4(a), and over the target area in Figure 4(b). Different initial perturbations are considered with a diagonal correlation (*D*), an autoregressive correlation (*AR*), and the superposition of autoregressive and TESV perturbations (*AR+SV*). *NON* denotes the non-assimilated ensemble (initialized with *AR+SV*). The first analysis (at 6 hours) has the highest impact on the quality of the solution. Different ensembles perform differently under data assimilation. The *AR* initialized ensemble gives slightly better solutions than the *D* initialized one. The *AR+SV* ensemble performs best over the target area and very well over the entire domain.



(a) Entire Domain (b) Target Area

Fig. 4. Time evolution of the ensemble standard deviation for different initial perturbations: diagonal correlation (D), autoregressive correlation (AR), and the superposition of autoregressive and TESV perturbations ($AR+SV$). NON denotes the non-assimilated ensemble (initialized with $AR+SV$). All other ensembles are analyzed every 6 hours using O_3 ground level observations in the target area

4 Conclusions

This paper discusses some of the challenges associated with the application of nonlinear ensemble filtering data assimilation to atmospheric chemical transport models. The distinguishing feature of these models is the presence of nonlinear and stiff chemical interactions occurring at very short characteristic time scales.

A correct initialization of the ensemble is necessary for a successful application of nonlinear filtering data assimilation. We propose to model background errors using multilateral autoregressive processes. Such models are computationally inexpensive and capture well the error correlations along the flow lines. Total energy singular vectors are calculated for chemically reactive flows. A dual projection technique (with the tangent linear and with the adjoint models) is proposed to keep the linearized solutions on the slow manifold and preserve the symmetry of the chemistry tangent linear – adjoint operators.

The data assimilation test problem considered here is based on a large scale simulation of air pollution in East Asia in March 2001. The ensembles are initialized using autoregressive models of background errors and total energy singular vectors. The superposition of these two types of initial perturbations leads to an ensemble that performs very well both over the target area and over the entire computational domain.

References

1. M. Buehner. Ensemble-derived stationary and flow-dependent background error covariances: Evaluation in a quasi-operational NWP setting. *Q.J.R.M.S.*, accepted, 2004.
2. G.R. Carmichael. STEM – A second generation atmospheric chemical and transport model. *URL: http:// www.cgrer.uiowa.edu*, 2003.

3. G.R. Carmichael et. al. Regional-scale chemical transport modeling in support of the analysis of observations obtained during the trace-p experiment. *J. Geophys. Res.*, pages 10649–10671, 2004.
4. W.P.L. Carter. Implementation of the SAPRC-99 chemical mechanism into the Models-3 framework. Technical report, United States Environmental Protection Agency, January 2000.
5. E.M. Constantinescu, A. Sandu, G.R. Carmichael, and T. Chai. Autoregressive models of background errors for chemical data assimilation. In preparation, 2005.
6. R. Daley. *Atmospheric Data Analysis*. Cambridge University Press, 1991.
7. H. Elbern, H. Schmidt, and A. Ebel. Variational data assimilation for tropospheric chemistry modeling. *J. Geophys. Res.*, 102(D13):15,967–15,985, 1997.
8. G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res.*, 99(C5):10,143–10,162, 1994.
9. G. Evensen. The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dyn.*, 53, 2003.
10. M. Fisher and D.J. Lary. Lagrangian four-dimensional variational data assimilation of chemical species. *Q.J.R.M.S.*, 121:1681–1704, 1995.
11. T.M. Hamill and J.S. Whitaker. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. *Mon. Wea. Rev.*, 129:2776–2790, 2001.
12. K.F. Hasselmann. Stochastic climate models. Part I. Theory. *Tellus*, 28:473–484, 1976.
13. P.L. Houtekamer and H.L. Mitchell. Data assimilation using an Ensemble Kalman Filter Technique. *Mon. Wea. Rev.*, 126(3):796–811, 1998.
14. P.L. Houtekamer, H. L. Mitchell, G. Pellerin, M. Buehner, M. Charron, L. Spacek, and B. Hansen. Atmospheric data assimilation with the ensemble Kalman filter: Results with real observations. *Mon. Wea. Rev.*, accepted, 2003.
15. A.H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.
16. R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Ser. D: J. Basic Eng.*, 83:95–108, 1960.
17. Lehoucq, R., K. Maschhoff, D. Sorensen, C. Yang. ARPACK Software (Parallel and Serial), http://www.caam.rice.edu/software/ARPACK.
18. W. Liao, A. Sandu, G.R. Carmichael, and T. Chai. Total energy singular vector analysis of atmospheric chemical transport models. Submitted, 2005.
19. R. Menard, S.E. Cohn, L.P. Chang, and P.M. Lyster. Stratospheric assimilation of chemical tracer observations using a Kalman filter. Part I: Formulation. *Mon. Wea. Rev.*, 128:2654–2671, 2000.
20. L. Menut, R. Vautard, M. Beekmann, and C. Honoré. Sensitivity of photochemical pollution using the adjoint of a simplified chemistry-transport model. *J. Geophys. Res.*, 105-D12(15):15,379–15,402, 2000.
21. L.P. Riishojgaard. A direct way of specifying flow-dependent background error correlations for meteorological analysis systems. *Tellus A*, 50(1):42–42, 1998.
22. A. Sandu, D. Daescu, and G.R. Carmichael. Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: I – theory and software tools. *Atm. Env.*, 37:5,083–5,096, 2003.
23. A. Sandu, Dacian N. Daescu, Gregory R. Carmichael, and Tianfeng Chai. Adjoint sensitivity analysis of regional air quality models. *J. Comp. Phys.*, accepted, 2004.
24. M. van Loon, P.J.H. Builtjes, and A.J. Segers. Data assimilation of ozone in the atmospheric transport chemistry model LOTOS. *Env. Model. Soft.*, 15:703–709, 2000.

Towards Dynamic Data-Driven Optimization of Oil Well Placement*

Manish Parashar[1], Vincent Matossian[1], Wolfgang Bangerth[2,4], Hector Klie[2], Benjamin Rutt[3], Tahsin Kurc[3], Umit Catalyurek[3], Joel Saltz[3], and Mary F. Wheeler[2]

[1] TASSL, Dept. of Electrical & Computer Engineering, Rutgers, The State University of New Jersey, New Jersey, USA
{parashar, vincentm}@caip.rutgers.edu
[2] CSM, ICES, The University of Texas at Austin, Texas, USA
{klie, bangerth, mfw}@ices.utexas.edu
[3] Dept. of Biomedical Informatics, The Ohio State University, Ohio, USA
{rutt, umit, kurc, jsaltz}@bmi.osu.edu
[4] Institute for Geophysics, The University of Texas at Austin, Texas, USA

Abstract. The adequate location of wells in oil and environmental applications has a significant economical impact on reservoir management. However, the determination of optimal well locations is both challenging and computationally expensive. The overall goal of this research is to use the emerging Grid infrastructure to realize an autonomic dynamic data-driven self-optimizing reservoir framework. In this paper, we present the use of distributed data to dynamically drive the optimization of well placement in an oil reservoir.

1 Introduction

The locations of wells in oil and environmental applications significantly affect the productivity and environmental/economic benefits of a subsurface reservoir. However, the determination of optimal well locations is a challenging problem since it depends on geological and fluid properties as well as on economic parameters. This leads to a very large number of potential scenarios that must be evaluated using numerical reservoir simulations. Reservoir simulators are based on the numerical solution of a complex set of coupled nonlinear partial differential equations over hundreds of thousands to millions of gridblocks. The high costs of simulation make an exhaustive evaluation of all these scenarios infeasible. As a result, the well locations are traditionally determined by analyzing only a few scenarios. However, this *ad hoc* approach may often lead to incorrect decisions with a high economic impact.

* The research presented in this paper is supported in part by the National Science Foundation Grants ACI 9984357, EIA 0103674, EIA 0120934, ANI 0335244, CNS 0305495, CNS 0426354, IIS 0430826, ACI-9619020 (UC Subcontract 10152408), ANI-0330612, EIA-0121177, SBR-9873326, EIA-0121523, ACI-0203846, ACI-0130437, CCF-0342615, CNS-0406386, CNS-0426241, ACI-9982087, CNS-0305495, NPACI 10181410, DOE ASCI/ASAP via grant numbers PC295251 and 82-1052856, Lawrence Livermore National Laboratory under Grant B517095 (UC Subcontract 10184497), Ohio Board of Regents BRTTC BRTT02-0003, and DOE DE-FG03-99ER2537.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 656 – 663, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Optimization algorithms offer the potential for a systematic exploration of a broader set of scenarios to identify optimum locations under given conditions. These algorithms, together with the experienced judgment of specialists, allow a better assessment of uncertainty and significantly reduce the risk in decision-making. Consequently, there is an increasing interest in the use of optimization algorithms for finding the optimum well location in oil industry [1,2,3,4]. However, the selection of appropriate optimization algorithms, the runtime configuration and invocation of these algorithms, and the dynamic optimization of the reservoir remains a challenging problem.

The overall goal of our research is to use the emerging Grid infrastructure [5] and its support for seamless aggregations, compositions and interactions, to enable the dynamic and autonomic data-driven optimization of oil reservoirs. In this paper we build on our autonomic reservoir management framework [6,7,8] to investigate the dynamic data-driven steering of the reservoir optimization processes for determining optimal well placement and configuration. The specific objective of this paper is to investigate how distributed data archives can be used to control and steer the optimization process to improve the quality as well as the speed of convergence.

2 Components of the Autonomic Data-Driven Oil Reservoir Framework

2.1 The Integrated Parallel Accurate Reservoir Simulator (IPARS)

IPARS represents a new approach to parallel reservoir simulator development, emphasizing modularity, code portability to many platforms, ease of integration and inter-operability with other software. It provides a set of computational features such as memory management for general geometric grids, portable parallel communication, state-of-the-art non-linear and linear solvers, keyword input, and output for visualization. A key feature of IPARS is that it allows the definition of different numerical, physical, and scale models for different blocks in the domain (i.e., multi-numeric, multi-physics, and multi-scale capabilities). A more technical description of IPARS and its applications can be found in [9].

2.2 Optimization Algorithms

We use different optimization algorithms in order to capture the complexities of the application. All of these algorithms need to be able to find the optimum very efficiently, i.e. with the least number of function evaluations while not requiring gradient information as that is generally unavailable in reservoir simulators.

Very Fast Simulated Annealing (VFSA): This algorithm is a variant of simulated annealing that speeds up the process by using a variable sampling algorithm that shrinks the sampling area as the temperature parameter is decreased. This allows for a more efficient local search towards the end of the optimization process. Additionally, we use different cooling schedules for the optimization variables, see [10].

Simultaneous Perturbation Stochastic Algorithm (SPSA): The SPSA algorithm is a gradient-based algorithm; however, instead of computing the exact gradient direction, it approximates it using a random stochastic direction. Consequently, it requires only two evaluations of the objective function per iteration, regardless of the dimension of the optimization problem. This allows for a significant decrease in the cost of optimization, especially in problems with a large number of decision parameters to be estimated. The algorithm is also suitable for noisy measurements of the objective function and can be customized to perform a more global search by injecting controlled random noise (e.g., see [11]).

Gradient based: This method approximates the gradient of the objective function to derive a search direction, and moves the present iterate along this direction. In practice, this algorithm is much less efficient than other methods [12], but we implement this standard algorithm for comparison since it is a very popular and widespread method.

2.3 Querying and Subsetting of Distributed Data: STORM

An increasingly important issue in Grid computing is to enable access to and integration of data in remote repositories. An emerging approach is the *virtualization* of data sources through relational and XML models [13–15]. STORM [16] is a service-oriented middleware that supports data select and data transfer operations on scientific datasets, stored in distributed, flat files, through an object-relational database model. In STORM, data subsetting is done based on attribute values or ranges of values, and can involve user-defined filtering operations. With an object-relational view of scientific datasets, the data access structure of an application can be thought of as a *SELECT* operation as shown in Figure 1. The <*Expression*> statement can contain operations on ranges of values and joins between two or more datasets. *Filter* allows implementation of user-defined operations that are difficult to express with simple comparison operations.

STORM services provide support to create a view of data files in the form of virtual tables using application specific *extraction* objects. An extraction object can be implemented by an application developer or generated by compiler [17]. It returns an ordered list of attribute values for a data element in the dataset, thus effectively creating a virtual table. The analysis program can be a data parallel program. The distribution of tuples in the parallel program is incorporated into our model by the *GROUP-BY-PROCESSOR* operation in the query formulation. *ComputeAttribute* is another user-defined function that generates the attribute value on which the selected tuples are grouped together based on the application specific partitioning of tuples.

STORM implements several optimizations to reduce the execution time of queries. These optimizations include 1) ability to execute a workflow through distributed filtering operations, and 2) execution of parallelized data transfer. Both data and task parallelism can be employed to execute filtering operations in a distributed manner. If a select expression contains multiple user-defined filters, a network of filters can be formed and executed on a distributed collection of machines. Data is transferred from multiple data sources to multiple destination processors in parallel by STORM data mover components.

SELECT *<Attributes>*
FROM $Dataset_1, Dataset_2, \ldots, Dataset_n$
WHERE *<Expression>* AND *Filter*(*<Attributes>*)
GROUP-BY-PROCESSOR *ComputeAttribute*(*<Attributes>*)

Fig. 1. Formulation of data retrieval steps as an object-relational database query

3 Autonomic Grid Middleware for Oil Reservoir Optimization

The autonomic Grid middleware supports interactions between application components, Grid services, resources (systems, CPUs, instruments, storage) and data (archives, sensors) [18]. It supports autonomic behaviors so that the interactions and feedback between simulations, services, sensors and data can be orchestrated using high-level rules, defined by expert, to navigate the parameter space and optimize the oil reservoir. Key components of the middleware are described below:

Discover [19] enables seamless access to, and peer-to-peer integration of applications, services, and resources on the Grid. The middleware substrate integrates Discover collaboratory services with the Grid services provided by the Globus Toolkit using the CORBA Commodity Grid (CORBACoG) Kit [20]. It also integrates the Pawn peer-to-peer messaging substrate [21]. Pawn enables decentralized (peer) services and applications to interact and coordinate over wide area networks. Finally, the DIOS [22] distributed object infrastructure that enables development and management of interactive objects and applications, encapsulating sensors and actuators, and a hierarchical control network. DIOS also allows the dynamic definition and deployment of policies and rules to monitor and control the behavior of applications and/or application services in an autonomic manner [23]. Detailed descriptions of the design, implementation, and evaluation of Discover components can be found in [19–23].

4 Integrated System for Data-Driven Oil Production Optimization

The oil production optimization process involves (1) the use of an integrated multiblock reservoir model and several numerical optimization algorithms (global and local approaches) executing on distributed computing systems on the Grid; (2) distributed data archives for historical, experimental (e.g., data from field sensors), and simulated data; (3) Grid services that provide secure and coordinated access to the resources and information required by the simulations; (4) external services that provide data, such as current oil market prices, relevant to the optimization of oil production or the economic profit; and (5) the actions of scientists, engineers and other experts, in the field, the laboratory, and in management offices.

In the process, item 1 is implemented by the IPARS framework. Both forward modeling (comparison of the performance of different reservoir geostatistical parameter scenarios) and inverse modeling (searching for the optimal decision parameters) can greatly benefit from integration and analysis of simulation, historical, and experimental data (item 2). Common analysis scenarios in optimization problems in reservoir simulations involve economic model assessment as well as technical evaluation

of changing reservoir properties (e.g., the amount of bypassed oil, the concentrations of oil and water). In a Grid environment, data analysis programs need to access data subsets on distributed storage systems [16]. This need is addressed by STORM. Figure 2 shows the performance of STORM for querying and subsetting seismic datasets. The performance numbers were obtained on a 30TB seismic dataset generated by simulations and stored on a 16-node disk-based cluster storage system, with 4.3GB/sec peak application-level bandwidth, at the Ohio Supercomputer Center. As seen from the figure, we can achieve close to 3.5GB/sec (about 75% of the peak bandwidth) bandwidth through runtime optimizations (such as distributed I/O, distributed filtering, multi-threading) implemented by STORM. The Discover autonomic Grid middleware provides the support for items 3, 4, and 5. We now discuss the use of Discover/Pawn to enable oil reservoir optimization [24].

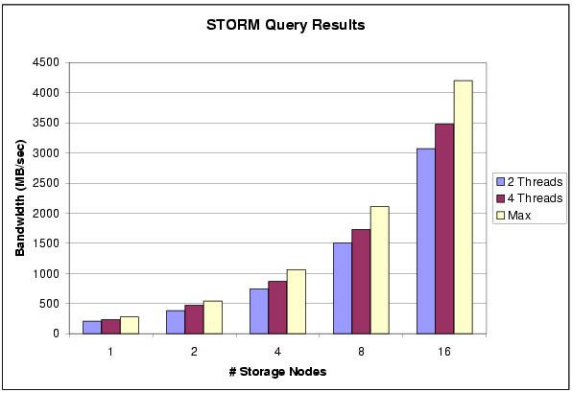


Fig. 2. Querying seismic data using STORM

The overall autonomic oil reservoir optimization scenario is illustrated in Figure 3. The peer components involved include: IPARS providing sophisticated simulation components that encapsulate complex mathematical models of the physical interaction in the subsurface, and execute on distributed computing systems on the Grid; IPARS Factory responsible for configuring IPARS simulations, executing them on resources on the Grid and managing their execution; Optimization Service (e.g. VFSA and SPSA); and Economic Modeling Service that uses IPARS simulation outputs and current market parameters (oil prices, costs, etc.) to compute estimated revenues for a particular reservoir configuration.

These entities dynamically discover and interact with one another as peers to achieve the overall application objectives. Figure 3 illustrates the key interactions involved: (1) The experts use pervasive portals to interact with the Discover middleware and the Globus Grid services to discover and allocate appropriate resource, and to deploy the IPARS Factory, Optimization Service, and Economic model peers. (2) The IPARS Factory discovers and interacts with the Optimization Service peer to configure and initialize it. (3) The experts interact with the IPARS Factory and Optimization Service to define application configuration parameters. (4) The Optimization

algorithm is seeded using DataCutter/STORM. This seed can be obtained by querying previously executed simulations. (5) The IPARS Factory then interacts with the Discover middleware to discover and allocate resources and to configure and execute IPARS simulations. (6) The IPARS simulation now interacts with the Economic model to determine current revenues, and discovers and interacts with the Optimization Service when it needs optimization. (7) The Optimization Service provides IPARS Factory with an improved well location, which then (8) launches new IPARS simulations with updated parameters. (9) Experts can at anytime discover, collaboratively monitor and interactively steer IPARS simulations, configure the other services and drive the scientific discovery process. Once the optimal well parameters are determined, the IPARS Factory configures and deploys a production IPARS run.

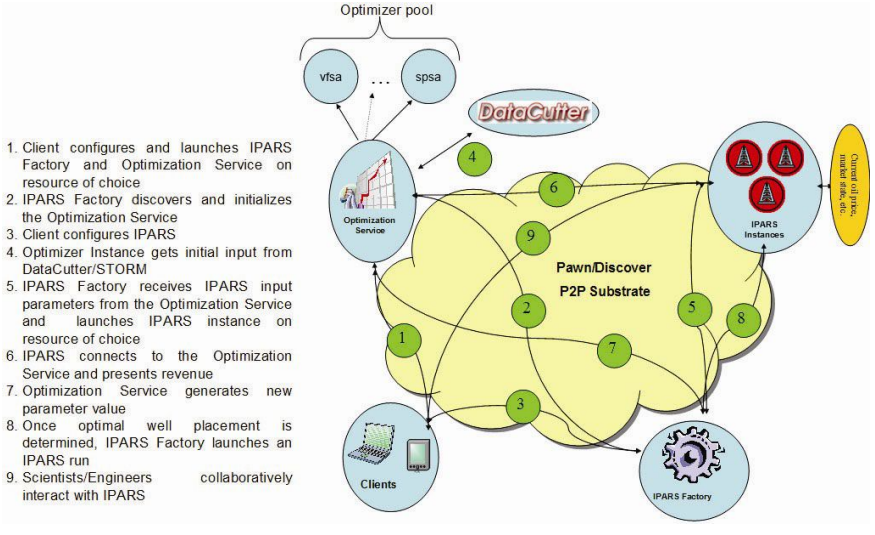


Fig. 3. Autonomic oil reservoir optimization using decentralized services

Figure 4 shows the progress of optimization of well locations using the VFSA and SPSA optimization algorithms for two different scenarios. The goal is to maximize profits for a given economic revenue objective function. The well positions plots (4(a) left and 4(b) right) show the oil field and the positions of the wells. Black circles represent fixed injection wells and a gray square at the bottom of the plot is a fixed production well. The plots also show the sequence of guesses for the position of the other production well returned by the optimization service (shown by the lines connecting the light squares), and the corresponding normalized cost value (4(a) right and 4(b) left).

The overall process described above is data-driven and autonomic in that the peers involved automatically detect sub-optimal oil production behaviors at runtime based on dynamically injected data, and orchestrate interactions among themselves to correct this behavior.

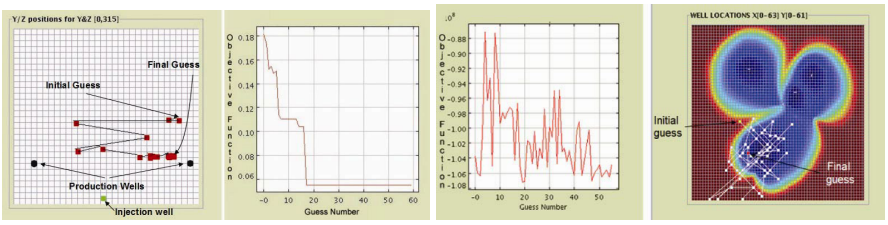


Fig. 4. Convergence history for the optimal well placement in the Grid using (a) VFSA algorithm and (b) SPSA algorithm

Further, the detection and optimization process is achieved using policies and constraints that minimize human intervention. The interactions between instances of peer services are opportunistic, based on runtime discovery and specified policies, and are not predefined.

5 Conclusion

We presented a novel infrastructure for enabling autonomic dynamic data-driven oil production management. We believe that such an infrastructure can aid in gaining better understanding of subsurface properties and decision variables, and can assist in the implementation of optimized oil production scenarios to lower infrastructure costs and maximize productivity.

References

1. Bittencourt, A.C., Horne, R.N.: Reservoir development and design optimization. In: SPE Annual Technical Conference and Exhibition, San Antonio, Texas (1997) SPE 38895.
2. Guyaguler, B., Horne, R.N.: Uncertainty assessment of well placement optimization. In: SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana (2001) SPE 71625.
3. Pan, Y., Horne, R.: Improved methods for multivariate optimization of field development scheduling and well placement design. In: SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana (1998) SPE 49055.
4. Yeten, B., Durlofsky, L.J., Aziz, K.: Optimization of nonconventional well type, location, and trajectory. SPE Journal **8** (2003) 200-210 SPE 86880.
5. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufman (2004).
6. Parashar, M., Klie, H., Catalyurek, U., Kurc, T., Matossian, V., Saltz, J., Wheeler, M.: Application of grid-enabled technologies for solving optimization problems in data-driven reservoir studies. The International Journal of Grid Computing: Theory, Methods and Applications (FGCS) **21** (2005) 19-26.
7. Matossian, V., Bhat, V., Parashar, M., Peszynska, M., Sen, M., Stoffa, P., Wheeler, M.F.: Autonomic oil reservoir optimization on the grid. Concurrency and Computation: Practice and Experience **17** (2005) 1-26.

8. Bangerth, W., Klie, H., Matossian, V., Parashar, M., Wheeler, M.F.: An autonomic reservoir framework for the stochastic optimization of well placement. Cluster Computing: The Journal of Networks, Software Tools, and Applications (2004) to appear.
9. IPARS: Integrated Parallel Reservoir Simulator, The University of Texas at Austin, http://www.ices.utexas.edu/CSM.
10. Sen, M., Stoffa, P.: Global Optimization Methods in Geophysical Inversion. Advances in Exploration Geophysics 4, editor: A.J. Berkhout. Elsevier (1995).
11. Spall, J.C.: Introduction to stochastic search and optimization, estimation, simulation and control. John Wiley & Sons, Inc., Publication, New Jersey (2003).
12. Bangerth, W., Klie, H., Wheeler, M.F., Stoffa, P.L., Sen, M.K.: On optimization algorithms for the reservoir oil well placement problem. Comp. Geosc., submitted (2004).
13. Open Grid Services Architecture Data Access and Integration. (*http://www.ogsadai.org.uk*)
14. Hastings, S., Langella, S., Oster, S., Saltz, J.: Distributed data management and integration: The mobius project. In: GGF Semantic Grid Workshop 2004. (2004) 20-38.
15. Li, X., Agrawal, G.: Using xquery for flat-file based scientific datasets. In: The 9th International Workshop on Data Base Programming Languages (DBPL). (2003).
16. Narayanan, S., Kurc, T., Catalyurek, U., Zhang, X., Saltz, J.: Applying database support for large scale data driven science in distributed environments. In: Proceedings of the Fourth International Workshop on Grid Computing (Grid 2003), Phoenix, Arizona (2003) 141-148.
17. Weng, L., Agrawal, G., Catalyurek, U., Kurc, T., Narayanan, S., Saltz, J.: An approach for automatic data virtualization. In: The Thirteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-13). (2004).
18. Agarwal, M., Bhat, V., Li, Z., Liu, H., Matossian, V., Putty, V., Schmidt, C., Zhang, G., Parashar, M., Khargharia, B., Hariri, S.: Automate: Enabling autonomic applications on the grid. In: Autonomic Computing Workshop, The Fifth Annual International Workshop on Active Middleware Services (AMS 2003), Seattle, WA USA (2003) 365-375.
19. Mann, V., Parashar, M.: Discover: A computational collaboratory for interactive grid applications. In Berman, F., Fox, G., Hey, T., eds.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons (2003) 727-744.
20. Parashar, M., von Laszewski, G., Verma, S., Gawor, J., Keahey, K., Rehn, N.: A CORBA Commodity Grid Kit. Concurrency and Computations: Practice and Experience **14** (2002) 1057-1074.
21. Matossian, V., Parashar, M.: Enabling peer-to-peer interactions for scientific applications on the grid. In Kosch, H., Boszormenyi, L., Hellwagner, H., eds.: Proceedings of the 9th International Euro-Par Conference. Volume 2790 of Lecture Notes in Computer Science, Springer-Verlag (2003) 1240-1247.
22. Muralidhar, R., Parashar, M.: A Distributed Object Infrastructure for Interaction and Steering. Special Issue - Euro-Par 2001, Concurrency and Computation: Practice and Experience **15** (2003) 957-977.
23. Liu, H., Parashar, M.: Dios++: A framework for rule-based autonomic management of distributed scientific applications. In Kosch, H., Boszormenyi, L., Hellwagner, H., eds.: Proceedings of the 9th International Euro-Par Conference. Volume 2790 of Lecture Notes in Computer Science, Springer-Verlag (2003) 66-73.
24. Matossian, V., Parashar, M.: Autonomic optimization of an oil reservoir using decentralized services. In: Proceedings of the 1st International Workshop on The Challenges for Large Applications in Distributed Environments (CLADE 2003), Computer Society Press (2003) 2-9.

High-Fidelity Simulation of Large-Scale Structures

Christoph Hoffmann, Ahmed Sameh, and Ananth Grama

Department of Computer Sciences, Purdue University, W. Lafayette, IN 47907

Abstract. Active structures capable of responding to external stimulii represent the emerging frontier in structural design. Robust and real-time sensing, control, and actuation pose fundamental challenges that must be addressed in this context. As part of an ambitious project funded by the National Science Foundation, researchers at Purdue, Rice, Florida State, and the Catholic (Belgium) Universities have undertaken development of these core technologies. Over the past 18 months, considerable progress has been made in the areas of model reduction and control, sensing, and simulation-based validation. This paper describes our results in high-fidelity simulations of large structures, subject to various (mechanical and thermal) stresses.

A high-fidelity simulation infrastructure poses a number of challenges. These include geometric modeling (generating a suitable mesh for the structure), physical modeling (developing mathematical models for coupling between various phenomena, specifying material properties), computational modeling (developing efficient numerical schemes and their parallel formulations), and appropriate visualization techniques. We have made fundamental contributions in each of these areas. Here, we overview some of our major contributions, along with sample simulations of existing structures. As part of our ongoing work, we also aim to perform a high-fidelity simulation of the tragic World Trade Center (WTC) crash. To this end, we have developed, from available blueprints, a highly resolved geometric model of the WTC. We also aim to complement all of our computational studies with detailed experimental validation on full-scale structures at the Bowen Lab for Structural Engineering. To the best of our knowledge, this is the first comprehensive effort to fully integrate simulation and modeling with sensing, control, and actuation in an experimental setting. In this sense, we believe that this project is a novel realization of the concept of dynamic data-driven application systems in the realm of large-scale structures.

1 Introduction

Physical processes are described primarily using mathematical models that are used to simulate the behavior of the underlying processes. Often, they are also used to modify or control system behavior. In this framework, there is an ever increasing need for improved accuracy, which leads to models of higher complexity. The basic motivation for system approximation is the need, in many instances,

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 664–671, 2005.
© Springer-Verlag Berlin Heidelberg 2005

for a simplified model of a dynamical system, which captures the main features of the original complex model. This need arises from limited computational, accuracy, and storage capabilities. The simplified model may then be used in place of the original complex model, either for *simulation*, or *control*. As sensor networks and embedded processors proliferate our environment, technologies for such approximations and real-time control emerge as the next major technical challenge.

As part of an ambitious NSF-funded project, we are exploring efficient and robust methods for producing reduced order models of large state-space systems, developing algorithms for real-time control, mechanisms for actuation, and systems infrastructure for affecting sensing, control, and actuation. To this end, we have made several fundamental contributions in the area of control [6, 5, 4, 3], sensing [1, 2], and modeling [7, 8].

A critical aspect of our study is the validation of underlying model reduction and control mechanisms using detailed computational simulations as well as experimental studies. The objectives of this are many-fold – we intend to precisely quantify errors in model reduction by simulating a hierarchy of progressively coarser models. To this end, we have developed a series of benchmarks, derived from a well-known structure (sanitized and shown in Figure 1). These benchmarks, derived from blueprints of the structure, have been made available over the public domain with the goal of comparing and validating results of control algorithms and model reduction. A second objective of developing a comprehesive simulation framework is to provide mechanisms for incorporating control algorithms and variable damping to study structural response. Finally, we have used our framework as a sophisticated diagnostic/prognostic tool for structural failure. Our work on analysis of the Pentagon crash (Figure 2) is widely accepted as a benchmark study in this regard.

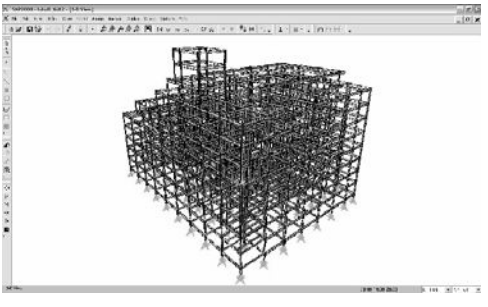
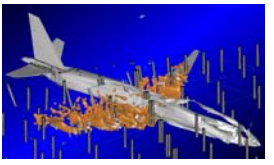


Fig. 1. Illustration of one of the benchmark structures used for model reduction and control

In this paper, we describe major technical challenges in high-fidelity simulations, our approach to addressing these challenges, and our recent results. We also describe ongoing work and the use of experimental studies to fully validate

(a) Aerial view of damage to the Pentagon building



(b) Simulated aircraft impact on RC columns in the building

Fig. 2. High fidelity simulation of the Pentagon crash performed in our group has yielded significant insights into structural properties of reinforced concrete and design of resilient buildings

our computational results. To the best of our knowledge, this is the first comprehensive effort to fully integrate simulation and modeling with sensing, control, and actuation in an experimental setting. In this sense, we believe that this is a novel realization of the concept of dynamic data-driven application systems in the realm of large-scale structures.

2 Technical Challenges

A number of technical challenges must be overcome to develop a comprehensive simulation framework. We have addressed these challenges and continue to explore more efficient techniques. We outline here some of our results in these areas.

2.1 Geometric Modeling – Meshing

Meshing represents one of the most time-consuming (labor intensive) part of the simulation process. The fidelity of a simulation is critically influenced by the mesh size and quality. A coarse discretization used for the elements of a Lagrangian mesh reduces the fidelity for contact and yielding behavior. Similarly, a coarse discretization used for the elements of an Eulerian mesh reduces the fidelity for mass transfer computations for the advection process, Euler-Lagrangian coupling, and the accuracy of fluid flow. In our simulation framework, we have created our own set of meshing tools and a detailed meshing methodology. This allows us to balance the desire for high resolution meshes, and the accuracy so obtained, with the necessity to accommodate the limitations of the underlying computing platform.

A completely automated meshing program that derives an FEA mesh from a geometric model is a significant undertaking. However, much can be accomplished when the meshing is semi-automatic. Since we typically combine separately meshed parts, we generate the mesh in two passes: the first pass generates a mesh description in an intermediate format that is independent of the underlying simulation engine. A second pass then translates the intermediate representation into a form that is suitable for input to selected simulation engines (in

our case, LS_Dyna). Thus, we can easily change the FEA package, for instance when comparing fidelity and performance of different FEA systems. We have extensive experience with this methodology and have successfully used it to generate meshes for our test structures, benchmark structures, as well as existing buildings (the Pentagon, Bank One building, and the World Trade Center).

2.2 Physical Modeling – Coupled Simulations

In addition to traditional issues of developing appropriate models for material behavior and determining suitable material properties, one of the major challenges arises from a coupling of multiple external stimulii. This happens, for instance, when structural failure is induced by thermal stresses. In such cases, it is necessary to simulate all of the physical processes in a tightly integrated manner. We show results from one such simulation performed by us in Figure 3. In this simulation of a fire on the 25th floor of the Bank One building in downtown Indianapolis, complete structural failure is caused by failure of a few critical beams. The underlying challenge is that the temporal and spatial discretizations associated with various phenomena potentially differ by several orders of magnitude. In the case illustrated here, the fire code used had a much finer spatial discretization and coarser temporal discretization than the structural counterpart. However, changes in structure critically impact progression of the fire – namely that changes in structure are critical for providing air for feeding the fire. A naive coupling of the two codes requires us to solve the Navier-Stokes equation for compressible fluid flow (air), the combustion model along with the

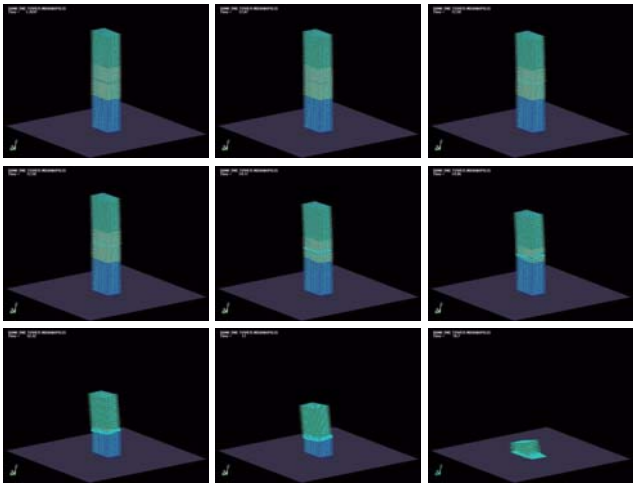


Fig. 3. Simulation of heat-induced structural failure – a fire on the 25th floor of the Bank-One tower in downtown Indianapolis is simulated using a coupled fire-structure code (wireframe model shown here). The entire simulation is available at http://www.cs.purdue.edu/homes/ayg/SIM

flow of the fuel, the structural model, and the solid body motion (for failed components), each at the finest granularity. This renders the simulation extremely computationally expensive.

The approach outlined about corresponds to an explicit timestepping scheme (which was also used in our simulation). While this scheme is generally easier to implement when using multiple production codes, stability in such schemes mandates the use of small timesteps. In contrast, implicit schemes permit use of larger timesteps, however, they require solution to a more complex system of equations. Solutions to such systems pose formidable challenges from a numerical standpoint. We are currently exploring a number of these research issues in the context of our simulation of the WTC crash (Section 3).

2.3 Computational Modeling – Numerical Methods and Parallelism

High-fidelity models, of the kind used in our simulations generate extremely large systems, which are typically solved iteratively. This puts the emphasis of time-to-solution on efficient numerical methods (preconditioners for iterative solvers) and their parallel formulations.

Our work on preconditioning linear systems arising from our structural simulations has resulted in extremely effective parallizable preconditioners. These preconditioners, based on our Spike [9] algorithm, are capable of fast convergence, low FLOP counts, and high concurrency. We have characterized the performance of solvers based on our preconditioner for a variety of meshes. We have studied the impact of various elements (eg., shell, beam, column) on the conditioning of the matrix and overall solution time. We demonstrate the superior performance of our methods in comparison to various existing solvers.

The size and complexity of systems resulting from our models necessitate the efficient use of powerful parallel platforms. For example, a single simulation instance of our Pentagon Crash, with one million nodes (3 d.o.f. per finite element node) over 0.25 seconds of real time takes over 68 hours on a dedicated 8 processor IBM Regatta SMP. A coarser model with 300K nodes over 0.2s of real time takes approximately 20 hours. In addition to the IBM SMP, we have also ported our codes to Linux/Itanium clusters. We have demonstrated excellent parallel efficiency for our codes for a variety of simulations. The computational requirements of these simulations provide strong motivation for effective error-bounded model reduction.

2.4 Visualization – The Human Interface

Large scale simulations produce massive multidimensional datasets. Although some scalars can be examined and presented using tables and graphs, 3D visualizations have long been used as powerful means of conveying such results. In the case of FEA simulations that analyze the mechanical interaction of entities under the extreme conditions of a high-kinetic-energy impact, visualization is an indispensable tool. In order to capitalize on high-fidelity finite element modeling, material and contact behavior, the visualization must have high fidelity as well.

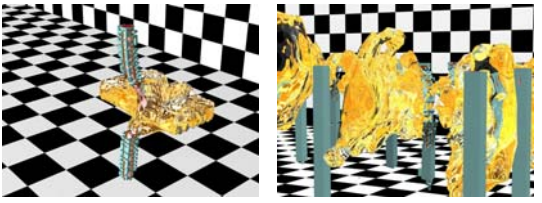


Fig. 4. Visualization of liquid, including reflection and refraction, for fuel flow in an aircraft crash

To affect a realistic and efficient visualization system, we build upon existing commercial solutions using plugins for data transfer, selection, material properties, and a variety of novel visualization techniques. We illustrate one such technique in Figure 4, which shows the flow of fuel through a set of columns. Such visualizations, developed in collaboration with experts in structural engineering, are very useful in understanding structural failure. More details of these techniques is provided in [7, 8].

3 Discussion and Ongoing Work

An immediate goal of our study is to perform high-fidelity simulations of the tragic World Trade Center (WTC) collapse, with a view to understanding, precisely, the structural cause of failure. We have obtained detailed blueprints of the structure and generated highly resolved geometric models (Figure 5). A major difficulty with this simulation is the coupling of thermal effects with mechanical stresses. This arises from the multitude of phenonema (fuel transport, fuel combustion including air-flow, combustibles in the building, mechanical impact, structural failure, etc.), timescales (all of these phenomena span over three orders of magnitude in time), and ambient factors (air-flow, temperature). Drawing on our initial simulation of the Bank-One building, and our experience in such simulations, we expect to demonstrate these simulations in the near-term.

A unique aspect of our study is that we aim to validate all of our computational results with detailed experimental studies. This effort relies heavily on extensive facilities available at the Bowen Lab for Structural Engineering at Purdue University. Leveraging other grants, we have built a full-scale three-storey test structure ($30 \times 50 \times 35$ feet), shown in Figure 6. A number of tests are planned on this structure, leading to the eventual testing to failure. These tests include low-frequency high-amplitude lateral and vertical loading (using hydraulic rams), low-amplitude tests for validating modes, and finally, impact of active damping on structural characteristics.

Our immediate goal in this area is to develop a real-time sensing network for measuring displacements, acceleration, and strain, at various points in the structure. For sensing displacements, we use laser sensors from Acuity Research (http://www.acuityresearch.com/contact-us.shtml). These sensors have a range

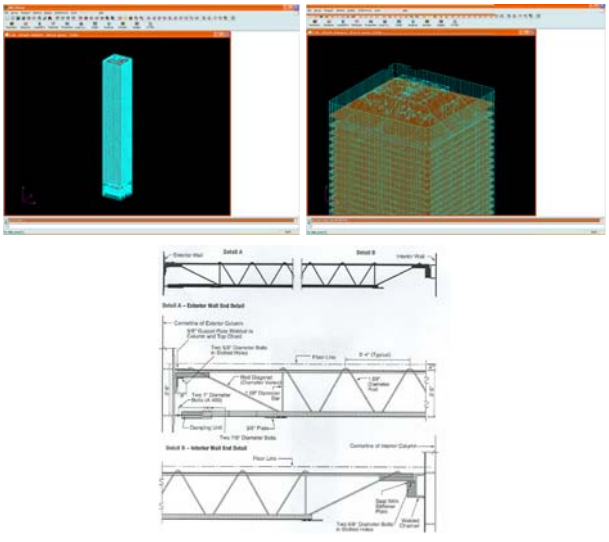


Fig. 5. A high-resolution mesh of the World Trade Center (WTC), developed using detailed blueprints (of the kind shown in bottom panel), which will be used for high-fidelity coupled fire-structure simulation of the WTC collapse

Fig. 6. Image of the test structure ($30 \times 50 \times 35$ feet) at the Bowen Lab for Structural Engineering at Purdue University. The structure will be subject to a variety of external loads and responses will be used to validate model reduction and simulation frameworks. A live web-cam image of this structure is posted at http://newton.ecn.purdue.edu/ ce/Bowen/Webcam/

from 11.5" to 27.5" with a resolution of 0.0048 inches at a frequency of 1250 Hz. These sensors provide RS232 outputs, which we network using RS232/BlueTooth interfaces. Accelerations are measured using the Crossbow Mica2 motes. These devices are networked to an XScale device, which is also equipped with suitable BlueTooth and 802.11b interfaces. The entire network is organized hierarchically

into a scalable, robust, and efficient network. Initial testing of this setup has been completed and the network is now being deployed on the test structure. In the process of this testing and deployment, we have also addressed a number of problems in sensor networking, including multiclass routing, resource-based routing, suitable operating system abstractions and programming interfaces, and power-aware resource management.

Acknowledgements

The authors would like to acknowledge Prof. Voicu Popescu and Dr. Sami Kilic for their help in generating some of the images in this paper. This work is supported by NSF contract ITR ACI-0324944.

References

1. B. Carbunar, A. Grama, and J. Vitek. Distributed and dynamic voronoi overlays for coverage detection and distributed hash tables in ad-hoc networks. *ICPADS*, 2004.
2. B. Carbunar, A. Grama, J. Vitek, and O. Carbunar. Coverage preserving redundancy elimination in sensor networks. In *Proceedings of the 1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Santa Clara, October 2004.
3. Y. Chahlaoui and P. Van Dooren. Benchmark examples for model reduction of linear time invariant dynamical systems. In P. Benner et al., editor, *Model Reduction of Dynamical Systems*. Springer Verlag, 2004.
4. Y. Chahlaoui and P. Van Dooren. Model reduction of time-varying systems. In P. Benner et al., editor, *Model Reduction of Dynamical Systems*. Springer Verlag, 2004.
5. Y. Chahlaoui, D. Lemonnier, A. Vandendorpe, and P. Van Dooren. Second-order balanced truncation. *Lin. Alg. Appl.*, 2005. to appear.
6. K. Gallivan, A. Vandendorpe, and P. Van Dooren. Model reduction of mimo systems via tangential interpolation. *SIAM J. Matrix Anal. Appl.*, 26(2):328–349, 2004.
7. C. Hoffmann, S. Kilic, V. Popescu, and M. Sozen. Integrating modeling, visualization and simulation. *IEEE Computating in Science and Engineering*, pages 52–60, January/February 2004.
8. C. Hoffmann and V. Popescu. Fidelity in visualizing large-scale simulations. *Computer-Aided Design*, 2005. to appear.
9. S.Kilic, F.Saied, and A.Sameh. Efficient iterative solvers for structural dynamics problems. *Computers & Structures*, 82(28):2363–2375, 2004.

A Dynamic Data Driven Grid System for Intra-operative Image Guided Neurosurgery*

Amit Majumdar[1], Adam Birnbaum[1], Dong Ju Choi[1], Abhishek Trivedi[2], Simon K. Warfield[3], Kim Baldridge[4,1], and Petr Krysl[2]

[1] San Diego Supercomputer Center, La Jolla, CA 92093, USA
[2] Structural Engineering Department, University of California San Diego, La Jolla, CA 92093, USA
[3] Computational Radiology Laboratory, Brigham and Women's Hospital, Harvard Medical School, Boston, MA 11111 USA
[4] Department of Chemistry, University of Zurich

Abstract. In the future, advanced biomechanical simulations of brain deformation during surgery will require access to multi-teraflop parallel hardware, supporting operating room infrastructure. This will allow surgeons to view images of intra-operative brain deformation within the strict time constraints of the surgical procedure - typically on the order of minutes, multiple times during a six or eight hour long surgery. In this paper we explore the grid infrastructure issues involved in scheduling, on-demand computing, data transfer and parallel finite element biomechanical simulation, which would guarantee that such a dynamic data driven real time application is actually feasible.

1 Introduction

Over the last decade, there has been tremendous progress toward fulfilling a very compelling idea: surgeons should be able to take advantage of imaging and computational technology to provide them with an enhanced ability to visualize complex hidden structures, even while operations are taking place [1-3]. Today, computer assisted image guided therapy (IGT) takes place only in research hospitals, supported by collaborative teams of clinicians, computer scientists and engineers. In this work, we explore some of the issues that must be tackled in order to fulfill the full promise of these prototype systems in the area of image-guided neurosurgery (IGNS).

The goal in neurosurgery is to provide 3D images of the brain that clearly delineate anatomical structures and tumor tissue. Key surgical challenges for neurosurgeons during tumor resection are to (1) remove as much tumor tissue as possible, (2) minimize the removal of healthy tissue, (3) avoid the disruption of critical anatomical structures, and (4) know when to stop the resection process. These challenges are compounded by the intra-operative shape deformation of the brain that happens as a result of tissue resection and retraction, injection of anesthetic agents, and loss of cerebrospinal fluid. The result is that the accuracy of the pre-operative plan diminishes steadily during the procedure.

* This research was supported in part by the NSF ITR grants CNS 0427183, 0426558, and NIH grants P41 RR13218, P01 CA67165, R01 LM0078651.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 672 – 679, 2005.
© Springer-Verlag Berlin Heidelberg 2005

It is therefore of great importance to be able to quantify and correct for these deformations while a surgery is in progress, by dynamically updating pre-operative images in a way that allows surgeons to react to changing conditions.

In our preliminary work [3] we have developed a prototype system, residing locally at the operating room at Brigham and Women's Hospital (BWH), that integrates data acquisition, quantitative monitoring, enhanced real-time control and intra-operative surgical navigation. Images of the patient's brain are captured pre-operatively and are registered against a brain anatomy atlas. During surgery, using an intra-operative scanner, new images of the patient's brain are acquired [4] and a finite element biomechanical model is solved [5] to construct a new visualization merging pre- and intra-operative data [6]. The time constraint on this overall procedure is severe and allows only about two minutes for the finite element biomechanical simulation.

The current prototype system at BWH uses a homogeneous, linear elastic biomechanical material model of the brain, primarily since only this crude and less accurate model is able to meet the real time constraint of IGT with the available dedicated on-site compute power at BWH. However, before the system is ready for broader adoption, the accuracy and precision for broader adoption, the accuracy and precision of the simulations will be significantly improved. Since the complex biomechanical model, under development, must be solved within the real time constraint of approximately two minutes, fast access to much larger-scale computational resources such as those found at the major national supercomputer centers, is required.

That said, the desire to harness remote multi-teraflop machines to enable improved results is at odds with a fundamental requirement of the system: absolute end-to-end *time criticality*. The implications of this requirement – on-demand, low latency, high bandwidth, reliable access to multi-teraflop parallel computers from the operating room – are the subject of this work.

2 Grid Infrastructure

The application described in this work sets requirements for "Grid" infrastructure [7,8], which would enable large-scale, remote access to distributed computational and data resources. The central focus of the original Grid infrastructure was the creation of standard protocols to enable uniform access to heterogeneous hardware platforms [9], embodied in the Globus toolkit [10]. Over the last several years, this strategy has shifted to one of integration with emerging enterprise distributed computing software systems, notably Web Services [11]. While progress has been made in these areas, the advances have generally been in the reduction of complexity in coordinating heterogeneous resources, perhaps at the cost of an increase in complexity of system administration and application development. In that context, the type of infrastructure development in the application described in this work may be considered as a driver of grid requirements.

A decade from now, we envision a worldwide network of computationally-assisted neurosurgery operating rooms (CAN-ORs). These CAN-ORs will include some combination of local compute clusters, as in our prototype at BWH, as well as remote, national-scale, shared computational infrastructure. The appeal of using local cluster

infrastructure is clear, as it sidesteps risks associated with contention for the remote shared resources. Unfortunately, a pure dependence on local resources sacrifices the greater precision and accuracy promised by the improved biomechanical models, discussed below, that require more substantial computational power. We are therefore attempting to delineate the broader infrastructure problems that must be solved in order to enable time-critical applications such as described here for the CAN-Ors.

The critical requirement of this application is the tight end-to-end time constraint. This requirement in turn places constraints on the performance and reliability of networks and the performance of computational resources. For large shared supercomputers, the dominant component of end-to-end performance is often that of queue wait time. Common supercomputing wisdom indicates that this queue delay depends on system policies, as well as on the size of the requested compute allocation. For example, it is common for systems to be configured to favor requests for large numbers of CPUs, but to also be able to "squeeze in" or "backfill" [12] jobs that require only a few CPUs and are short in duration.

The Teragrid [13] is an NSF funded grid infrastructure across multilple academic and research sites within the US. We measured queue delays on two of the Teragrid clusters over a three day period, for requests of 5 minutes of wallclock time, from 2 to 64 CPUs. We submitted only a single job at a time; if the job had not started within the critical timeframe of 10 minutes, the submission was terminated, and the next request processed. The purpose of this measurement was to illustrate the likelihood that the finite element simulations would be able to run on a Teragrid cluster under our stringent time constraints. In all, we submitted 313 jobs to the NCSA Teragrid cluster, and 332 to the SDSC Teragrid cluster, which represents from 50 to 56 jobs of each size, on each cluster.

Figure 1 shows the percentage of tasks that advanced through the queue successfully, and started to execute before the 10-minute time limit elapsed. As one might predict, both clusters show a decreasing likelihood of success with increasing number of requested CPUs. This decline is more dramatic for the NCSA machine, which is larger and thus currently more in demand than the SDSC cluster. Figure 2 shows the mean total queue delay for the tasks that did make it through the queue successfully. Again, there appears to be a direct relationship between the size of request and the length of the queue delay, although it would also appear that the two clusters exhibit different performance profiles. The behavior of queuing systems clearly merits further study, including more rigorous statistical characterization based on much larger data sets.

These results just illustrated depict clearly the challenge this application faces. Even for very modest resource requests, the likelihood that CPUs will become available within the critical time period is very low. One possible alternate solution would be to simply submit each job to many distributed resources. We prototyped such a "flooding" scheduler, that submits a single job to many resources; as soon as the job runs on one resource, it is removed from all of the other queues. This approach has worked well in our early tests and demonstrations. However, this technique would be disastrous if used on a large scale, as it would undermine the ability for batch schedulers to make reasonable choices. Furthermore, the issue of data transfer would become even more onerous if we had even less ability to predict where the job would run. What is really required is a scheduler that can infer the likelihood of success on various resources, again a subject for future work.

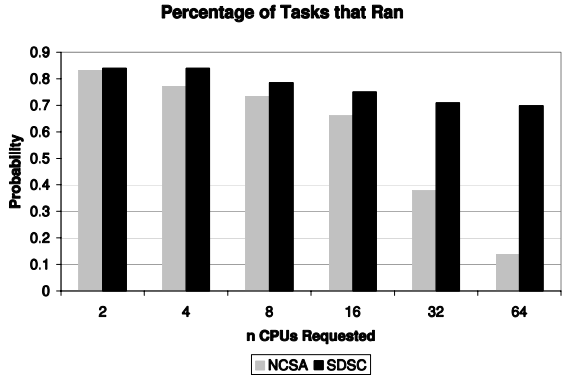


Fig. 1. Percentage of submitted tasks that ran, as a function of CPUs requested

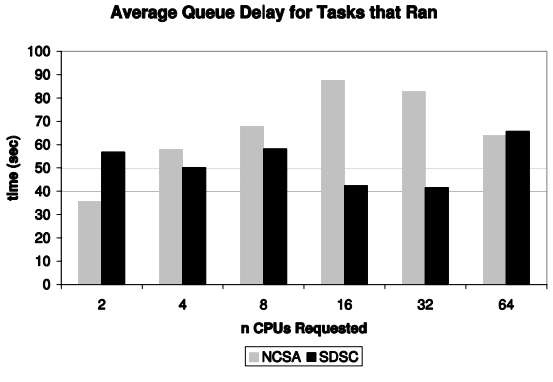


Fig. 2. Average queue delay for tasks that begun running within 10 minutes

2.1 Network and Data Transfer

Access to remote supercomputers implies the need for reliable transfer of input and output data sets. Of course the most desired solution would be to use a dedicated network with guarantee Quality of Service (QoS), however the reality is that the network will be shared, so our system will ultimately need to detect and react to a variety of changing conditions. Also, as we envision a scenario where there are 100s of CANORs sending time critical requests for simulation to say 10s of supercomputer centers across the nation, the expectation of a dedicated network may not be a realistic solution. We have performed preliminary tests of network speed for transfer of 20MB file (this is the size of the volumetric brain deformation result from the current finite element model) between SDSC's TeraGrid [13] machine and a machine residing inside the firewall at the BWH hospital. Note that BWH, as with every hospital that must protect patient privacy, is behind a firewall, that in this case only allows a single port of entry to the internal network.

We have investigated also Storage Resource Broker (SRB) [14] as a means to transfer and share data between BWH and SDSC. The SRB is client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network. The BWH has acquired a SRB rack as a part of their research collaboration with the Biomedical Informatics Research Network (BIRN) [15]. The SRB server at BWH exists outside the firewall and enables a direct feed of data onto the BWH SRB from the SDSC TeraGrid cluster; this means that a machine inside the firewall must retrieve the data from the BWH SRB server.

We also measured file transfer rates using the globus-url-copy [10] which demonstrates slower performance going into BWH compared to coming out of BWH. This is again due to the firewall, which allows only one port of entry into BWH but allows multiple ports to go out of BWH firewall. The following table compares the results for the transfer of a 20 MB file using globus-url-copy, SRB and SCP (with two different sets of command line options.)

Table 1. Time to Transfer 20 MB file

Transfer Direction	globus-url-copy	SRB	scp	scp –C
TG to BWH	50	49	68	31
BWH to TG	9	12	40	30

3 Advanced Biomechanical Model Development

Our finite element simulation model, currently under development, is based on the conforming hierarchical adaptive refinement method (CHARMS) [16,17]. Inspired by the theory of wavelets, this refinement method produces globally compatible meshes by construction. The framework based on CHARMS avoids mesh compatibility problems, and provides extended support for object oriented design. CHARMS based solver FAMULS [18] provides an ideal choice for time critical applications. The present objective is to replicate results produced by the existing crude biomechanical finite element solver, currently in use by the prototype system at BWH, and prove the effectiveness of the new solver, called FAMULS which has the additional capability of adaptive mesh refinement. Figure 3 shows simulation of a deformation case with the FAMULS elastic solver using adaptive mesh refinement which added additional tetrahedra in the interior to reduce error. The color indicates deformation in the direction of y axis. This case reproduces the deformation that was originally calculated by the existing crude finite element model.

The current solver is based on the small strain isotropic elastic principle. However, in order to expand the capacity of the system to predict brain deformation ahead in time beyond the deformation shown by the intraoperative 3D volumetric MRI, the accuracy and precision of the simulations must be improved. We are working to replace the current biomechanical model with an anisotropic non-homogeneous visco

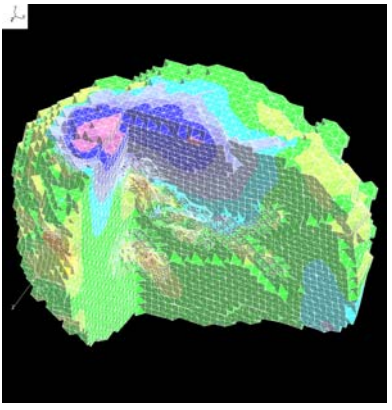


Fig. 3. Mesh cutoff showing displacement contour in color

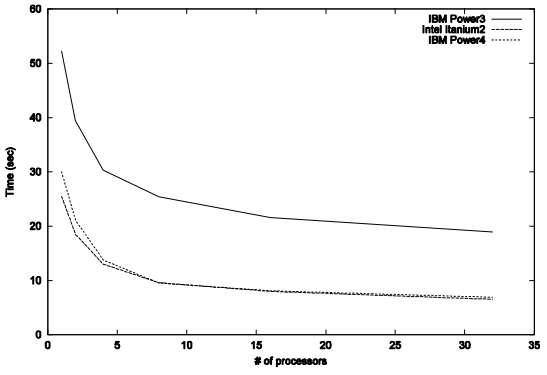


Fig. 4. Performance of the Current Linear Elastic Finite Element Model

elastic and visco plastic, scalable, nonlinear model which will give more accurate predictions. Since this complex model still has to meet the real time constraint of neurosurgery, it requires fast access to much larger computational resources than those typically located in a hospital operating room. This motivates our efforts to create a novel grid architecture for real-time data driven simulations during image guided neurosurgery.

In order to plan our deployment onto a heterogeneous Grid platform, we have started testing performance of the current linear elastic biomechanical model on various parallel computers. We have ported the existing parallel code to the IBM Power3 (8-way node; 375 Mhz; 4 GB memory/node), IBM Power4 (8-way node; 1.5 Ghz; 16 GB memory/node), and Intel Itanium2 clusters (2-way node; 1.5 Ghz; 4 GB memory/node), exemplifying typical heterogeneous platforms that would be available for grid computing involving our application.

Figure 4 shows computation time required for the current linear elastic biomechanical simulation model on these three different parallel machines. In this simulation the total number of meshes was 43584, defining 214035 tetrahedral elements.

The Figure clearly indicates the importance of focusing on improved scalability as we develop the improved model, and further directs our attention to machines with fast internal network fabrics.

4 Conclusions

In the long term, the contribution we are attempting to make is to clarify the requirements for Grid infrastructure to support time-critical medical applications such as IGNS. This infrastructure, which may consist perhaps of hundreds of operating rooms and tens of computational resource centers, will be based on improved networking and software infrastructure. In this paper, we have reported our initial progress, and have discussed some areas in which further progress is required.

The described research involves development and deployment of an integrated and practical grid architecture for focused, on demand computer-assisted neurosurgery. The initial working prototype system currently operating at BWH, using a homogeneous, linear elastic biomechanical material model of the brain, is being improved both in terms of algorithmic enhancements as well as grid support infrastructure for more reliable processing and response time.

Our future plans include the continued development of the advanced biomechanical parallel finite element model based on FAMULS. The parallel scaling of this advanced model will also be improved such that the simulation can be performed within the defined time constraint on a multi-teraflop parallel machine. With respect to grid-computing issues, our objectives include the rigorous characterization of the queue wait time of various classes of jobs on many production clusters, as well as issues related to network traffic performance to these geographically distributed clusters from BWH. These two metrics will allow us to predict, with certain confidence, the overall end-to-end time needed for this time critical application of IGNS.

References

1. D.L.G. Hill, J.V. Hajnal, D. Rueckert, S.M. Smith, T.Hartkens, and K.McLeish. A Dynamic Brain Atlas. *IEEE Trans. Medical Imaging*, 18, Issue 8:712--721, 1999.
2. Y. Kawasaki, F. Ino, Y. Mizutani, N. Fujimoto, T. Sasama, Y. Sato, N. Sugano, S. Tamura, and K. Hagihara. High-Performance Computing Service Over the Internet for Intraoperative Image Processing. *IEEE Transaction on Information Technology in Biomedicine*, 8, No. 1:36--46, 2004.
3. S.K. Warfield, F. Talos, A. Tei, A. Bharatha, A. Nabavi, M. Ferrant, P.M. Black, F.A. Jolesz, and R. Kikinis. Real-Time Registration of Volumentric Brain MRI by Biomechanical Simulation of Deformation During Image-Guided Neurosurgery. *Journal of Computing and Visualization in Science*, 5:3--11, 2002.
4. P.M. Black, T. Morairty, E. Alexandar, P. Stieg, E. J. Woodard, P. L. Gleason, C.H. Martin, R. Kikinis, R.B. Schwartz, and F. A. Jolesz. The Development and Implementation of Intra-operative MRI and its Neurosurgical Applications, *Neurosurgery*, 41:831--842, 1997.
5. S.K. Warfield, F. Jolesz, and R. Kikinis. A High Performance Computing Approach to the Registration of Medical Imaging Data. *Parallel Computing*, 24:1345--1368, 1998.

6. M. Ferrant, A Nabavi, M. Macq, F. A. Jolesz, R. Kikinis, and S.K. Warfield. Registration of 3D Intraoperative MR Images of the Brain Using a Finite Element Biomechanical Model. *IEEE Transactions on Medical Imaging*, 20, Issues 12:1384--1397, 2001.
7. Ian Foster and Carl Kesselman, editors. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., 1999.
8. Fran Berman, Geoffrey Fox, and Anthony J. G. Hey. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Inc., 2003.
9. Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1--??, 2001.
10. http://www.globus.org.
11. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.
12. D. Lifka. The anl/ibm sp scheduling system. In D.G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 949 of Lecture Notes in Computer Science, pages 295--303. Springer-Verlag, 1995.
13. http://www.teragrid.org.
14. http://www.sdsc.edu/srb/.
15. Biomedical Informatics Research Network (BIRN), http://www.nbirn.net.
16. P. Krysl, A. Trivedi, and B. Zhu. Object Oriented Hierarchical Mesh Refinement with CHARMS. *International Journal of Numerical Methods in Engineering*, Vol. 60, Issue 8:1401--1424, 2004.
17. J. D. Mcnamara. Health Monitoring of Rail Road Tracks by Elastic Wave Based Monitoring Technique. *Ph.D Thesis* , Structural Engineering Department, University of California San Diego, 2004.
18. http://hogwarts.ucsd.edu/~pkrysl/software.html.

Structure-Based Integrative Computational and Experimental Approach for the Optimization of Drug Design*

Dimitrios Morikis[1], Christodoulos A. Floudas[2], and John D. Lambris[3]

[1] Department of Chemical and Environmental Engineering, University of California, Riverside, CA 92521, USA
dmorikis@engr.ucr.edu
[2] Department of Chemical Engineering, Princeton University, Princeton, NJ 08544, USA
floudas@titan.princeton.edu
[3] Department of Pathology and Laboratory Medicine, University of Pennsylvania, Philadelphia, PA 19104, USA
lambris@mail.med.upenn.edu

Abstract. We present an integrative approach for the optimization in the design of peptides which are candidates to become therapeutic agents. This approach is based on the structure of the peptide ligand when structural information on the protein target is not available. Our approach combines (i) NMR spectroscopy, (ii) structure determination by distance geometry, simulated annealing, and global optimization methods, restrained with NMR-derived or deduced restraints, (iii) molecular dynamics simulations, based on NMR low energy, averaged minimized, or ensemble of structures, (iv) in silico sequence selection using integer linear optimization, (v) fold specificity using deterministic global optimization, and (vi) peptide synthesis, mass spectrometry characterization, and activity measurements. The optimization of the design of the 13-residue cyclic peptide compstatin is presented as a paradigm for the application of our approach. The same principles can be applied for the design of small proteins with desired properties and function.

1 Introduction

We present an overview of our integrative approach for peptide-drug discovery. This approach is based on the peptide structure and it incorporates classical and novel features for drug design methodologies. The approach integrates several methods, computational and experimental, for structural analysis, structure determination, sequence and structure prediction, and structure-activity correlations, including spectroscopy, spectrometry, molecular dynamics simulations, integer linear optimization and global optimization, and binding constant and IC_{50}-value activity

* This work was supported by grants from NIH and NSF.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 680 – 688, 2005.
© Springer-Verlag Berlin Heidelberg 2005

measurements. We will present brief descriptions for the use and integration of the computational and experimental methods. We will follow with a specific example for the application of our approach on the optimization of the peptide compstatin, a potent inhibitor against the abnormal (or unregulated) function of the complement system (part of innate immunity).

Our approach can be used for the optimization in the design of active peptide analogs, with the aim to improve their activity. Optimally, the structures of both the ligand peptide and the target protein or the active (binding) site of the target protein are desirable. In this paper we focus on ligand-based design, using the structure of the peptide when the structure of the protein is not available. This is often the case for large or unstable proteins, which are not yet amenable to structure determination by crystallography (X-ray or neutron diffraction) or NMR methods.

2 Methods

Figure 1 presents the interplay of the computational and experimental methods used in our approach. The methods used in the various steps will be briefly described below.

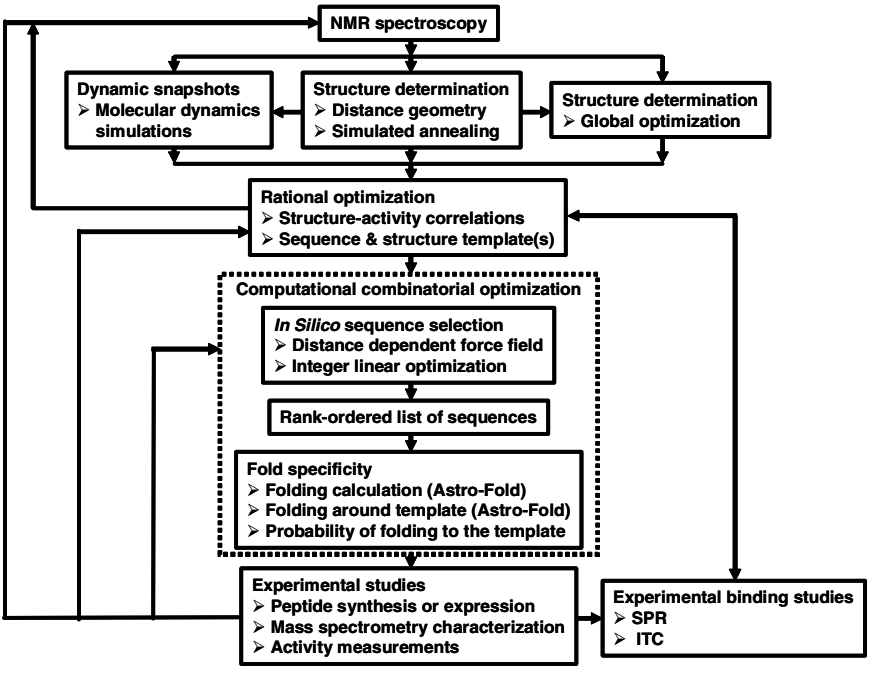


Fig. 1. Flow of information in our integrative drug design and optimization approach

2.1 NMR Spectroscopy

Multi-dimensional and multi-nuclear NMR spectroscopy is used to determine the secondary structure and tertiary or quaternary contacts of peptides and proteins [1]. Typically, two-dimensional NMR spectra are sufficient for structural analysis of peptides. In cases of overlapping cross peaks three-dimensional heteronuclear NMR spectra may be necessary to increase spectral resolution. Heteronuclear NMR spectra require, in most cases, ^{15}N- and or ^{13}C-labeled samples, prepared by expression rather than chemical synthesis methods.

The measured NMR parameters are chemical shifts, spectral linewidths and heights, areas, or volumes, coupling constants, and nuclear Overhauser effects (NOEs). The chemical shifts are used to assign specific resonances for protons and ^{13}C and ^{15}N. Patterns of chemical shifts are used to identify the amino acid systems. Deviations of chemical shifts in structured peptides from their random coil values, plotted as differences or chemical shift indices, are used to distinguish helical, beta-strand, and random coil secondary structures. Temperature variation of chemical shifts often points to the formation of hydrogen bonded secondary structure, when plotted in the form of temperature coefficients. In certain triple resonance NMR spectra, connectivities of chemical shifts are used to identify the protein backbone by piecing together the backbone N, C_α, C' atoms, and side chain C_β atoms. Coupling constants are also used to distinguish well-formed helical from beta-strand secondary structures. NOEs are used to determine through space dipolar interactions of protons in proximity of less than 5.5-6 Å. Specific sequential and medium range NOE connectivity patterns are used to determine secondary structure and to distinguish helices, strands, and turns, from extended or random conformations. Long range NOEs are used to identify tertiary and inter-molecular contacts in the case of complexes. Spectral linewidths are used to determine relaxation parameters and to identify aggregation or binding.

2.2 Structure Determination Using NMR Restraints

Distances and backbone and side chain torsion angles are typically used as experimental restraints, together with restraints from chemical knowledge for covalent geometry and nonbonded interactions, in computational methods for the determination of three-dimensional structures of peptides [2]. Distances are derived from NOEs and torsion angles are derived from J-coupling constants, sometimes in combination with NOEs. The structure determination protocols are based on the minimization of an energy function using parameters and topologies from a specific force field, in the Cartesian or torsion angle space.

Distance Geometry and Molecular Dynamics-Based Simulated Annealing. Distance geometry methods are based on the metric matrix theorem which allows for the conversion of distances among a set of points in three-dimensional Euclidian space into Cartesian coordinates. Because not all atom pair distances are

known in the metric matrix, large distances need to be assigned and subsequently reduced using the triangle inequality. In some instances, the distance geometry method is the first step in structure determination followed by simulated annealing regularization of the structures. Alternatively, simulated annealing methods are used alone.

Molecular dynamics-based simulated annealing for the determination of three-dimensional structures using NMR restraints is a simplified implementation of regular (unrestrained) molecular dynamics *in vacuum* (without the presence of explicit solvent molecules or implicit solvation energy term) [2]. The potential energy function has the form $E=E_{covalent}+E_{nonbonded}+E_{experimental}$, where $E_{covalent}=E_{bond}+E_{angle}+E_{torsion}+E_{chiral,planar}$, and $E_{experimental}=E_{distance}+E_{torsion}+(E_{chemicalshift}+E_{couplingconstant})$. The terms in parentheses are optional or entered during the later stages of the calculations. $E_{covalent}$ and $E_{nonbonded}$ are empirical energy terms from known chemical principles. $E_{nonbonded}$ is typically a simplified repel potential term that accounts for van der Waals and electrostatic interactions of nonbonded atom pairs. In certain instances or at later stages in the calculations, deduced hydrogen bonds can be used in the form of distance restraints. The structure determination protocols are iterative procedures involving the stepwise addition of restraints and tests for the correctness of the assigned restraints.

Deterministic Global Optimization. Global optimization methods have been tested first for the structure determination of peptides using NOE restraints [3]. The structure determination formulation represents a general nonconvex constrained global optimization problem, a class of problems for which several methods have been developed. In this work, the formulations are solved via the αBB deterministic global optimization approach, a branch and bound method applicable to the identification of the global minimum of nonlinear optimization problems with twice-differentiable functions [4]. The global minimization of a detailed atomistic energy force field $E_{forcefield}$ is performed over the set of independent torsion angles, which can be used to describe any possible configuration of the system. The bounds on the torsion angles are enforced by simple box restraints. Finally, a set of distance constraints, which are nonconvex in the internal coordinate system, are used to constrain the system. The energy function has the form $E_{forcefield}=E_{electrostatic}+E_{vanderWaals}+E_{Hbond}+E_{torsion}$.

2.3 Molecular Dynamics Simulations

Molecular dynamics simulations using implicit solvent representation are performed to scan rapidly the conformational space of peptides and proteins [1]. The initial structures are the complete ensemble of NMR-derived structures, the averaged minimized structure, or the lowest energy structure. The empirical energy function has the form $E=E_{covalent}+E_{nonbonded}+E_{solvation}$, where $E_{covalent}$ includes the covalent geometry energy terms $E_{bond}+E_{angle}+E_{torsion}+E_{chiral,planar}$, $E_{nonbonded}$ includes the nonbonded interaction terms $E_{vanderWaals}+E_{electorstatic}+(E_{Hbond})$, and $E_{solvation}$ is an appropriate impicit solvation model energy term. Simulation times in the range of 1-

10 ns are typically sufficient for peptides. Energy minimizations are performed before and after the molecular dynamics simulations.

Individual snapshots during a trajectory are analyzed to examine (i) flexibility using the RMSD of backbone and side chain atoms and calculated B-factors, (ii) secondary structure using backbone torsion angles and hydrogen bonding, and (iii) tertiary structure using side chain contacts. Pairwise nonbonded interactions between side chains are determined by plotting the total potential energy, the van der Waals energy, and electrostatic energy. This type of analysis evaluates the role of side chains in packing and stability. Molecular dynamics also provide free energy differences of different conformations and motional amplitudes for conformational inter-conversion.

2.4 Rational Design and Optimization

Rational design and optimization depends on the availability of three-dimensional structures [5,6]. In rational design the structures or structural information of active analogs from NMR and computational methods (see above) are used to identify the critical for binding and activity physico-chemical properties and their spatial arrangement. This is done in combination with experimental binding and activity measurements (see below) by determining structure-(dynamics-binding)-activity correlations. The parentheses denote often optional steps; however these steps are necessary in our opinion because in several cases the lowest energy structure derived from NMR or crystallography is not the one that binds. It is not unusual for the ligand or the protein active site to undergo structural rearrangements upon binding. Structure-(dynamics-binding)-activity correlations are used to optimize the peptide by rationally replacing or modifying its building blocks. Ligand-based Pharmacophore and QSAR (quantitative structure-activity relationship) approaches use rational design for the construction of pharmacophore models. Peptide pharmacophore models can be used to identify matching low molecular mass organic molecules from appropriate databases. It is usual in rational design to incorporate non-natural amino acids or amino acids substituted with specific chemical groups (e.g., methylation, etc), which need parameterization for the computational methods described below.

2.5 Computational Combinatorial Design and Optimization

A two-stage method for the de novo peptide and protein design has been recently introduced. This method is based on sequence and structural templates of active analogs, determined from NMR- or crystallographically-derived structures, or from snapshots of molecular dynamics trajectories. The first step of the method involves sequence selection and the second step involves fold validation.

Sequence Selection. The sequence selection step relies on a constrained integer linear programming (ILP) model [7,8]. Sequences compatible with given sequence and backbone templates are identified by minimizing an empirical potential describing

pairwise distance-dependent interactions. This potential assigns energy values for amino acid interactions, based on the C_α-C_α separation distance for each amino acid pair. The type of amino acids (side chains) is implicitly included in the interaction potentials. The solution of this ILP problem allows the identification of a rank ordered list of the low lying energy sequences, which are used in the second step, the prediction of fold stability and specificity, described next.

Prediction of Fold Specificity. This step is used to distinguish the most optimal sequences from those identified in the sequence selection step (above) according to rigorous quantification of conformational probabilities [7,8]. Conformational ensembles are generated for the selected sequences under two sets of conditions. First, the structure is allowed to vary around the template structure, with imposed fluctuations. The fluctuations can be based on the structural boundaries defined by the NMR ensemble, on the RMSDs of molecular dynamics structures, or some arbitrary fluctuation. Second, a free folding calculation is performed with limited number of restraints, as needed (e.g., disulfide bridges, etc), but with the underlying template structure not being enforced. The ensembles of conformers resulting from the two folding calculations are used to calculate the probabilities. The formulation of the folding calculations is similar to the structure determination calculations using deterministic global optimization (described above). The calculations are performed with ASTRO-FOLD framework [9].

2.6 Experimental Peptide Synthesis and Binding and Activity Measurements

Sample preparation is typically performed using solid state peptide synthesis or expression methods and tested for integrity using mass spectrometry. Biological assays for activity measurement in the form of IC_{50} values are used to assess the strength of the peptide inhibitors. The IC_{50} value is the peptide concentration at 50% inhibition. Direct inhibition or competition assays using ELISA methods are used, depending on the specifics of the experiment. Binding data using surface plasmon resonance (SPR) and isothermal titration calorimetry (ITC) methods are often necessary to correlate structure with binding and activity and to form hypotheses [1].

3 Results: The Example of Compstatin

The 13-residue peptide compstatin is an inhibitor of the complement system, with sequence I[CVVQDWGHHRC]T-NH_2, where brackets denote cyclization through a disulfide bridge [5]. The complement system is part of innate immunity and acts together with adaptive immunity to fight infection by foreign pathogens. The complement system is finely regulated by fluid phase or cell-bound regulators and is programmed to recognize “self” from “non-self”. When this regulation brakes down,

as is the case in several autoimmune diseases and pathological situations, the complement system turns against "self" and attacks host tissues. Currently there is no drug in the clinic that regulates the abnormal complement activation.

Compstatin was identified by constructing and testing a phage-displayed random peptide library against binding to complement component C3 [10], a converging component of the complex pathways of complement activation. The three-dimensional structure of compstatin was determined using NMR data and computational methods restrained with NMR-derived restraints. The computational methods were hybrid distance geometry/simulated annealing [11] and global optimization [3]. Subsequently, optimization of the sequence of compstatin was performed using rational design based on NMR structural studies (but not computational complete structure determination) and structure-activity correlations, which yielded several active analogs with up to 4-fold higher inhibitory activity than the parent peptide [6]. The rational design determined that 7 of the 13 amino acids of compstatin were indispensable for activity, and provided the following sequence template for further optimization: Ac-**X**[CV**X**QDWG**XXX**C]**X**-NH_2 (called active sequence template), where the 6 amino acids marked with **X** were optimizable [5]. The active sequence template was used for the construction of a second round of phage-displayed peptide library and binding experiments against C3 (called experimental combinatorial optimization). This round of optimization yielded four more active analogs, with one of them being 4-fold more active than the parent peptide [12]. The active sequence template and the NMR-derived structure of compstatin were also used as the first test case of the novel computational combinatorial approach described above, which predicted several active analogs [7,8]. Among the active analogs that were synthesized and experimentally tested for activity were 5 analogs with 6- to 14-fold higher inhibitory activities than the parent peptide [7,8,13]. Subsequent rounds of rational design, using the same active sequence template and NMR-based structural studies identified several active analogs with up to 99-fold higher inhibitory activities than the parent peptide [13]. Several of these analogs are peptidomimetics because they are built using combinations of natural and non-natural amino acids. Table 1 shows the major breakthroughs in the optimization of the design of compstatin, from each of the various optimization rounds.

Besides IC_{50} activity measurements that were performed for each synthesized analog, kinetic and thermodynamic binding studies were performed using surface plasmon resonance [12,14], and isothermal titration calorimetry [15]. The structural, binding, and activity studies were useful to form testable structure-binding and structure-activity hypotheses.

Finally, quasi-dynamic pharmacophore models have been generated using snapshots from molecular dynamics simulations of several active and inactive compstatin analogs [16,17]. Upon selection of proper geometric and physico-chemical properties to represent the spatial arrangement of the pharmacophore points, one model allowed for the distinction of active from inactive analogs [17].

Table 1. Benchmarks in the optimization of the design of compstatin*

Peptide	Sequence	RIA	Year	Ref.
I	I[CV V QDWGHHRC]T-NH_2	1	1996	10
II	**Ac**-I[CV V QDWG**A**HRC]T-NH_2	4	2002	6
III	**Ac**-**L**[CV V QDWG**W**HRC]**G**-NH_2	4	2003	12
IV	**Ac**-I[CV **Y** QDWG**A**HRC]T-NH_2	14	2003	7,8
V	**Ac**-I[CV **W** QDWG**A**HRC]T-NH_2	45	2005	13
VI	**Ac**-I[CV(**2Nal**)QDWG**A**HRC]T-NH_2	99	2005	13

*RIA, relative inhibitory activity. Bold face indicates additions/substitutions responsible for the increase in inhibitory activity. Ac, acetylation; 2Nal, 2-naphthylalanine.

4 Conclusions

We have described a ligand-based approach for the optimization of active peptides, which are candidates to become therapeutics. This approach involves the interplay of a variety of computational and experimental methods. The compstatin example presented here, has demonstrated the effectiveness of our ligand-based approach in the absence of structural information for the C3 target, the structure of which thus far has not been determined. We expect that similar mixed computational and experimental approaches will also be useful when the structures of both, the free ligand and the free target, and/or their complex, are available.

References

1. Morikis, D., and Lambris, J.D.: Physical methods for structure, dynamics and binding in immunological research. Trends Immunol. 25 (2004) 700-707
2. Güntert, P.: Structure calculation of biological macromolecules from NMR data. Quart. Rev. Bioph. 31 (1998) 145-237
3. Klepeis, J.L., Floudas, C.A., Morikis, D., Lambris, J.D.: Predicting peptide structures using NMR data and deterministic global optimization. J. Comp. Chem. 20 (1999) 1354-1370
4. Floudas, C.A.: Deterministic global optimization: theory, methods and applications, Kluwer Academic Publishers, Dordrecht, The Nederlands (2000)
5. Morikis, D., Soulika, A.M., Mallik, B., Klepeis, J.L., Floudas, C.A., and Lambris, J.D.: Improvement of the anti-C3 activity of compstatin using rational and combinatorial approaches. Biochem. Soc. Trans. 32 (2004) 28-32
6. Morikis, D., Roy, M., Sahu, A., Troganis, A., Jennings, P.A., Tsokos, G.C., and Lambris J.D.: The structural basis of compstatin activity examined by structure-function-based design of peptide analogs and NMR. J. Biol. Chem. 277 (2002) 14942-14953
7. Klepeis, J.L., Floudas, C.A., Morikis, D., Tsokos, C.G., Argyropoulos, E., Spruce, L.A., Lambris, J.D.: Integrated computational and experimental approach for lead optimization and design of compstatin variants with improved activity. J. Am. Chem. Soc. 125 (2003) 8422-8423
8. Klepeis, J.L., Floudas, C.A., Morikis, D., Tsokos, C.G., and Lambris, J.D.: Design of peptide analogues with improved activity using a novel de novo protein design approach. Ind. Eng. Chem. Res. 43 (2004) 3817-3826

9. Klepeis, J.L. and Floudas, C.A.: ASTRO-FOLD: a combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. Biophys. J. 85 (2003) 2119-2146
10. Sahu, A., Kay, B.K., Lambris, J.D.: Inhibition of human complement by a C3-binding peptide isolated from a phage-displayed random peptide library. J. Immunol. 157 (1996) 884-891
11. Morikis, D., Assa-Munt, N., Sahu, A., Lambris, J.D.: Solution structure of compstatin, a potent complement inhibitor. Protein Sci. 7 (1998) 619-627
12. Soulika, A.M., Morikis, D., Sarrias, M.R., Roy, M., Spruce, L.A., Sahu, A., Lambris, J.D.: Studies of Structure-Activity Relations of Complement Inhibitor Compstatin. J. Immunol. 171 (2003) 1881-1890
13. Mallik, B., Katragadda, M., Spruce, L.A., Carafides, C., Tsokos, C.G., Morikis, D., and Lambris J.D.: Design and NMR characterization of active analogs of compstatin containing non-natural amino acids. J. Med. Chem. 48 (2005) 274-286
14. Sahu, A., Soulika, A.M., Morikis, D., Spruce, L.A., Moore, W.T., and Lambris, J.D.: Binding kinetics, structure-activity relationship, and biotransformation of the complement inhibitor Compstatin. J. Immunol. 165 (2000) 2491-2499
15. Katragadda, M., Morikis, D., and Lambris, J.D.: Thermodynamics studies on the interaction of the third complement component and its inhibitor, compstatin. J. Biol. Chem. 279 (2005) 54987-54995
16. Mallik, B., Lambris, J.D., Morikis, D.: Conformational inter-conversion of compstatin probed with molecular dynamics simulations. Proteins 53 (2003) 130-141
17. Mallik, B. and Morikis D.: Submitted (2005)

Simulation and Visualization of Air Flow Around Bat Wings During Flight

I.V. Pivkin[1], E. Hueso[2], R. Weinstein[2],
D.H. Laidlaw[2], S. Swartz[3], and G.E. Karniadakis[1]

[1] Division of Applied Mathematics
[2] Department of Computer Science
[3] Department of Ecology and Evolutionary Biology,
Brown University, Providence, RI 02912, USA

Abstract. This paper presents a case study of interdisciplinary collaboration in building a set of tools to simulate and visualize airflow around bat wings during flight. A motion capture system is used to generate 3D coordinates of infrared markers attached to the wings of a bat flying in a wind tunnel. Marker positions that cannot be determined due to high wing deformation are reconstructed on the basis of the proper orthogonal decomposition (POD). The geometry obtained for the wings is used to generate a sequence of unstructured tetrahedral meshes. The incompressible Navier-Stokes equations in arbitrary Lagrangian-Eulerian formulation are solved using the hybrid spectral/hp element solver Nektar. Preliminary simulation results are visualized in the CAVE, an immersive, 3D, stereo display environment.

1 Introduction

Fluid dynamics approaches have revolutionized our understanding of insect flight, revealing aerodynamic mechanisms almost unimaginable only 25 years ago [1–9]. Although bat and bird flight is also likely to yield insights of use in future technological application, such as the development of unmanned micro-air vehicles [10,11], researchers of vertebrate flight are just beginning to incorporate in their work sophisticated methodologies drawn from the physical and mathematical sciences [12−14]. The unique features of bats – their specialized skeletal anatomy, high muscular control over wing conformation, and highly deformable wing-membrane skin – yield wings that undergo large changes in 3D geometry with every wing-beat cycle, and consequently produce highly maneuverable and energetically efficient flight [15−18].

To date, however, bat flight has not been studied from the quantitative perspective of unsteady aerodynamics. In particular, there is no rigorous understanding of the mechanisms by which the bat generates the high aerodynamic coefficients necessary for its flight capabilities, or of the vortex structure associated with the bat's exquisite aerodynamic control. Simulation of airflow around wings that not only flap but also undergo enormous shape changes in each wing-beat poses significant technical challenges, as does visualizing the complex 3D data such studies necessitate.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 689 – 694, 2005.
© Springer-Verlag Berlin Heidelberg 2005

This paper describes our interdisciplinary collaboration to build a set of tools for simulation and visualization of airflow around bat wings during flight. The next section gives a brief overview of our data acquisition techniques. In section 3 we discuss data preprocessing, which is necessary to provide input for the numerical simulations described in section 4. Finally, the visualization of preliminary simulation results is presented in section 5.

2 Data Acquisition

The motion-capture data of bat flight were acquired by flying more than 20 individuals of several species through a wind tunnel [18]. Two high-speed digital cameras tracked infrared markers attached to the bat wings. We chose to study the flight of a Pteropus Policephalus because of its large size and relatively slow motion.

After selecting 160 video frames where the motion of the bat is close to one complete wing beat, the Peak [19] motion capture system was used to extract the 3D coordinates of markers. These data are utilized to animate a simple 3D polygonal model of the surface of the bat's wings [20] (Fig. 1). Due to high wing deformations during flight, the 3D coordinates of a few markers cannot be determined in some frames. At the heart of generating polygonal bat wing model is the ability to reconstruct accurately the positions of these markers.

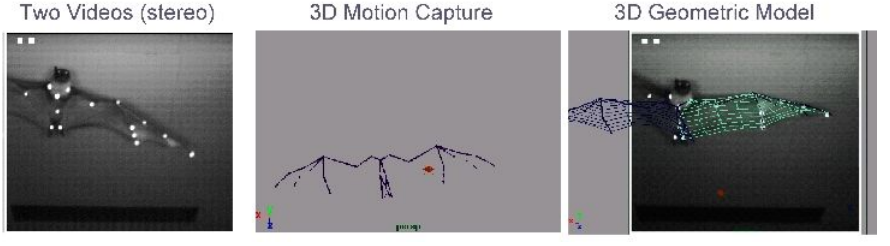


Fig. 1. Video capture of bat wing shape changes

3 Missing Data Reconstruction

Here we follow an approach based on proper orthogonal decomposition (POD) combined with the least-squares approach first proposed in [21] for image reconstruction. Let us consider a vector of data points $\mathbf{u}(t) \in (\Re^3)^N$ given for each frame t, where N is the number of markers attached to the bat wings. We assume that we have available a finite number P of frames. We then look for a representation of $\mathbf{u}(t)$ in the form

$$\mathbf{u}(t) = \boldsymbol{\varphi}_0 + \sum_{k=1}^{N} \alpha_k \boldsymbol{\varphi}_k, \tag{1}$$

where $\boldsymbol{\varphi}_0 = \sum_{t=1}^{P} \mathbf{u}(t)$ is a mean (time-averaged) position of the markers and $\boldsymbol{\varphi}_k$ are the orthonormal spatial modes. The unknown functions $\boldsymbol{\varphi}_k$ can be calculated by minimizing an energy functional, producing a POD or Karhunen-Loeve decomposition.

The formulation assumes the completeness of the data; modifications are required if there exist a space-time regions in which components of $\mathbf{u}(t)$ are missing or corrupted. The procedure proposed in [21] completes the missing data iteratively starting from the average value at missing data location as the initial guess for the unknowns.

In order to make the bat motion cyclic, as is desirable for numerical simulations, we enforce periodicity on the POD mode coefficients $\alpha_k(t)$.

4 Numerical Simulations

The animated polygonal model is the basis for a sequence of tetrahedral meshes of the volume of 10 by 10 by 20 around the bat geometry, which has a wing span of approximately two non-dimensional units at its widest. Wings are represented by an infinitely thin tessellation of triangles (Fig. 2). An arbitrary Lagrangian-Eulerian (ALE) formulation of the incompressible Navier-Stokes equations is employed to solve for the flow field. This allow us to run simulations with changing geometry without remeshing each time step. A single tetrahedral mesh can be deformed to fit a number of frames, typically between 7 and 15 depending of the rate of deformation. When the deformation of the mesh becomes too extreme, elements degenerate and a new tetrahedral mesh must be created. As a result, multiple meshes are necessary and must be interpolated together in order to simulate an entire wing-beat. The mesh generator Gridgen [22] is used to generate up to 15 meshes for one wing-beat. Each mesh has approximately 6000 spectral tetrahedral elements. The governing equations are solved using the hybrid spectral/hp element solver Nektar [23].

Preliminary simulations were performed using third-order polynomial expansion in each element. The Reynolds number was set to 100, greatly reducing computational

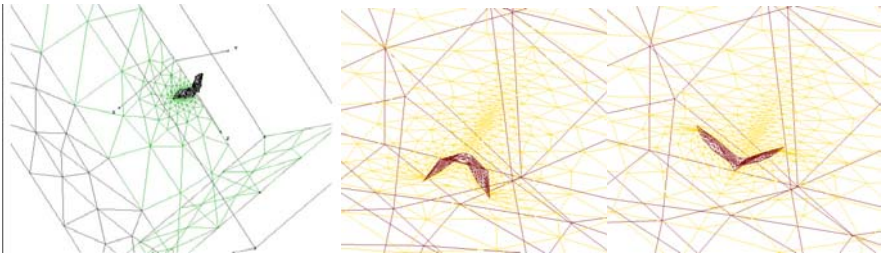


Fig. 2. Some of the meshes used in simulations

effort and time. The solver produced 40 snapshots over one periodic motion cycle that describe the fluid velocity and the pressure distribution around the animated bat wings.

5 Visualization

The size and complexity of the time varying fields generated by the simulation makes them unfit for real-time visualization. To overcome this limitation, we pre-compute and store sets of pathlines and streamlines that can later be visualized interactively. The line sampling and visualization methods used on the bat flow data is a variation of the one presented in [24] for the visualization of blood flow in a coronary artery.

We visualize the flow data in the CAVE [25], an immersive 3D stereo display environment that scientists find more engaging than our less sophisticated desktop displays. One of our visualizations shows massless particles flowing down pre-computed pathlines that resemble eels of variable length, color and opacity. A second visualization relies on animated streamlines, represented with lines of variable color and opacity. The user has interactive control over the number of lines displayed, how randomly they are distributed in space, and the mapping of opacity to flow quantities. These controls allow users to explore a continuum between localized visualization of detected vortices and the contextual flow around them [26] (Fig. 3).

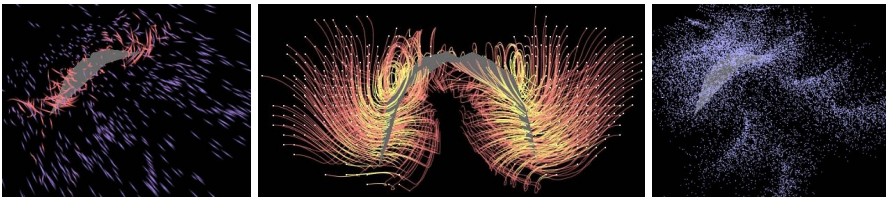


Fig. 3. Three different visualization methods are used to show different characteristics of the structure of the simulated flow around a motion-captured bat: (left to right) particle eels display pathlines, time-varying streamlines show vortices atop the wing during a down beat, and white dots capture structures in the wake

6 Discussion and Conclusions

Our studies differ from previous studies of bat flight in an important way: we do not treat the wings as simple oscillating plates, but instead explicitly characterize the changing intrinsic wing structure of potentially great aerodynamics importance. This makes realistic Reynolds number simulations challenging. The preliminary results reported here were obtained for Re=100, too low to make judgments about the mechanisms by which the bat executes its flight capabilities. Even with higher Reynolds number simulations, the results would ideally be compared with

experimental measurements to validate our process. Our current simulation results let us develop prototype visualization techniques that we believe can be extended to the flows with higher, more realistic Reynolds numbers. However, we have found that the preprocessing time and complexity of the visualization tools make them difficult to use on an everyday basis. Initial feedback does show that these tools help significantly in the exploration of complex time-dependent flow data.

In conclusion, we have completed a full iteration of simulation and visualization of unsteady flow around the bat wings during the flight. We believe that progress in understanding the relationship among wing shape, movement, and airflow during bat flight benefits significantly from the kind of interdisciplinary collaboration presented in this paper.

Acknowledgements

This work was supported in part by NSF (CCR-0086065 and CNS-0427374).

References

1. M.H. Dickinson and K.G. Gotz. Unsteady aerodynamic performance of model wings at low Reynolds numbers. *Journal of Experimental Biology*, 174:45–64, 1993.
2. M. H. Dickinson, F.O. Lehmann, and S.P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.
3. M.H. Dickinson. Unsteady mechanisms of force generation in aquatic and aerial locomotion. *American Zoologist*, 36(6):537–554, 1996.
4. R. Dudley. *The biomechanics of insect flight: form, function, evolution*. Princeton University Press, 2000.
5. C.P. Ellington. The aerodynamics of hovering insect flight. IV. Aerodynamic mechanisms. *Phil. Trans. Roy. Soc. Lond. B*, 305(1122):1–15, 1984.
6. C.P. Ellington. Limitations on animal flight performance. *Journal of Experimental Biology*, 160:71–91, 1991.
7. C. P. Ellington. Leading-edge vortices in insect flight. *Nature*, 384(6610):626–630, 1996.
8. S. Vogel. Flight in drosophila. II. Variations in stroke parameters and wing contour. *J. Exp. Biol.*, 46:383–392, 1967.
9. A.P. Willmott, C.P. Ellington, and A.L.R. Thomas. Flow visualization and unsteady aerodynamics in the flight of the hawkmoth, manduca sexta. *Philosophical Transactions of the Royal Society of London Series B – Biological Sciences*, 352(1351):303–316, 1997.
10. C.P. Ellington. The novel aerodynamics of insect flight: Applications to micro-air vehicles. *Journal of Experimental Biology*, 202(23):3439–3448, 1999.
11. G.R. Spedding and P.B.S. Lissaman. Technical aspects of microscale flight systems. *Journal of Avian Biology*, 29(4):458–468, 1998.
12. J.M.V. Rayner, P.W. Viscardi, S. Ward, and J.R. Speakman. Aerodynamics and energetics of intermittent flight in birds. *American Zoologist*, 41:188–204, 2001.
13. G.R. Spedding, A. Hedenstrom, and M. Rosen. Quantitative studies of the wakes of freely flying birds in a low-turbulence wind tunnel. *Experiments in Fluids*, 34(2):291–303, 2003.
14. B.W. Tobalske, T.L. Hedrick, K.P. Dial, and A.A. Biewener. Comparative power curves in bird flight. *Nature*, 421:363–366, 2003.

15. Y. Winter, C. Voigt, and O. Von Helversen. Gas exchange during hovering flight in a nectar-feeding bat glossophaga soricina. *Journal of Experimental Biology*, 201(2):237–244, 1998.
16. C.C. Voigt and Y. Winter. Energetic cost of hovering flight in nectar-feeding bats (phyllostomidae : Glossophaginae) and its scaling in moths, birds and bats. *Journal of Comparative Physiology B – Biochemical Systemic and Environmental Physiology*, 169(1):38–48, 1999.
17. E.F. Stockwell. Morphology and flight manoeuvrability in new world leaf-nosed bats (chiroptera : Phyllostomidae). *Journal of Zoology*, 254:505–514, 2001.
18. Sharon Swartz, K. Bishop, and M.F. Ismael-Aguirre. Dynamic complexity of wing form in bats: implications for flight performance. In *Functional and evolutionary ecology of bats*. Oxford Press, 2005. In Press.
19. Peak performance, http://www.peakperform.com.
20. R. Weinstein, I. Pivkin, S. Swartz, D.H. Laidlaw, G. Karniadakis, and K. Breuer. Simulation and visualization of air flow around bat wings during flight. Technical Report CS-02-16, Brown University Computer Science Department, August 2002.
21. R.M. Everson and L. Sirovich. The Karhunen-Loeve transform of incomplete data. *Journal of the Optical Society of America, A*, 12(8):1657, 1995.
22. Gridgen, http://www.pointwise.com.
23. G.E. Karniadakis and S.J. Sherwin. *Spectral/hp Element Methods for CFD*. Oxford University Press, 1999.
24. J. Sobel, A. Forsberg, D.H. Laidlaw, R. Zeleznik, D. Keefe, I. Pivkin, G. Karniadakis, P. Richardson, and S. Swartz. Particle flurries: Synoptic 3D pulsatile flow visualization. *IEEE Computer Graphics and Applications*, 24(2):76–85, March/April 2004.
25. C. Cruz-Neira, D.J. Sandin, and T.A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Proceedings of ACM SIGGRAPH*, volume 27, pages 135–142. ACM, August 1993.
26. E. Hueso. Visualizing vortices in simulated air flow around bat wings during flight. Technical Report CS-03-25, Master's Project, Computer Science Department, Brown University, Providence, RI, December 2003.

Integrating Fire, Structure and Agent Models*

A.R. Chaturvedi[1], S.A. Filatyev [2], J.P. Gore[2], A. Hanna[1],
J. Means[1], and A.K. Mellema[1]

[1] Purdue Homeland Security Institute, Purdue University, West Lafayette, IN 47907
[2] School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907

Abstract. A state-of-the-art integrated environment was created to study interaction among fire, structure and agent models in a fire evacuation from a typical office building. For the fire simulations NIST large-eddy simulation code Fire Dynamics Simulator (FDS) was used. The code is based on a mixture fraction model. FDS provided time resolved temperature, CO, CO2, soot distribution in the building. The agent software was designed to simulate agent behaviors during evacuation by tracking the behavior of each individual in the building taking into account effects of temperature, CO, and soot on the behavior and health for each agent. The created integrated environment was designed to provide the bridge between multiple simulations for data transfer and model interaction. It was shown that fire position, agent positions, and number of exits available affect significantly agents' health and death toll. The results can be used for better fire safety building design and regulations.

1 Introduction

As the complexity of buildings increases, it becomes more and more challenging to provide a satisfactory level of fire safety in the buildings. Even if the building satisfies modern fire safety codes, it does not necessarily guarantee needed safety levels to the occupants.

The events of September 11, 2001 showed necessity of better response training in case of emergencies such as terrorist attack on a big building. In order to prepare for such an event, an understanding of the event is required. Such understanding can come from scientific analysis of human/environmental interaction. In case of explosion or arson in a building, such as that studied in the present paper, the important aspects are fire propagation, human interaction with the fire, interaction between the fire and the building structure. Realistic models describing fire, structure response, and human behavior are needed for such simulations, as well as an environment where those models run simultaneously and interact with each other during execution. Currently, models exist which simulate fire, human behavior during evacuation and building structure, but these models have yet to be integrated to simulate real-time events and decisions [1].

* This research is partially funded by NSF ITR grant # CNS-0325846.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 695 – 702, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Goal

The goal of the project is to create a state-of-the-art integrated environment to study interaction between fire, structure and agents in emergencies situations. Realistic and complex models of fire, structure, and agent were used in order to simulate realistic behavior. These three simulations are normally run independently which created the need for a medium to transfer information from one simulation to another. In order to do this the shared reality environment was created.

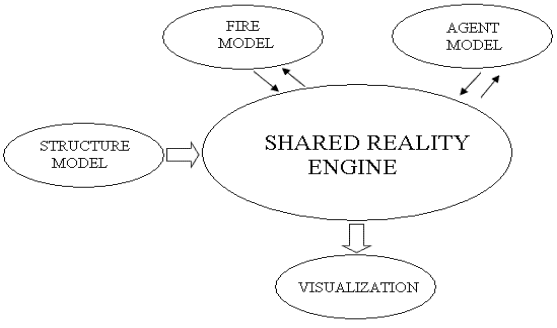


Fig. 1. Shared reality concept

3 Data Format

The original building layout was produced in 3D Studio Max. The format needed to be changed for the needs of fire, structure, and agent models since each program was developed independently and uses different input formats. The conversion was done in the shared reality engine [1,2]. At each second of the scenario all three models calculated changes and decisions based on input describing events that took place during the previous second. The output of one model was obtained by the shared reality engine, converted to the formats for the other models and transferred to them. All outputs were then collected by the shared reality engine and converted to text format which was read by the 3D visualization used for analysis. The visualization allowed for the presentation of the impact of certain emergencies, to find the correct measured response and, if possible, to adjust modifications to building plans.

4 Models

The original scenario was evacuation from an office building floor in case of fire with realistic fire model simulations and agent simulation capturing individual agent behavior.

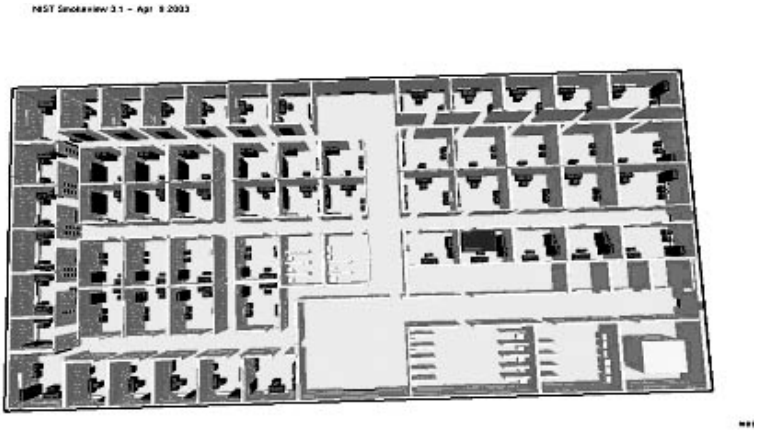


Fig. 2. Plan of the floor used in both fire and agent simulations

4.1 Geometry

The geometry used in calculations represented a typical floor of an office building. Physical dimensions were 50m by 30m by 3m; the building layout is shown in Figure 2. The building had windows and doors open to the environment during simulations at all times. Office doors were also open at all times. Almost every office along the perimeter had an open window. There were five exits, three along the West wall (left), one main exit on the South wall (bottom), and one exit on the North wall (top). Changes in the structure of the building were calculated using LS-DYNA [3,4].

4.2 Fire Model

Fire was simulated using the Fire Dynamics Simulator (FDS 4) program developed by NIST [5]. FDS 4 is a large-eddy simulation (LES) turbulent code which employs the mixture fraction model. The code has been successfully used to describe different processes from fundamental combustion studies of plumes [6] to the replication of 2001 fire of the World Trade Center [7].

For the current simulations all walls, the ceiling and the floor were assumed to have flammable properties. It was assumed that the fire started as a result of arson. To accomplish this numerically a tank of kerosene was placed in one of the rooms and ignited. The tank is shown as a gray rectangle in one of the rooms on the right side of Figure 2. This room has exits into a corridor separating the floor into almost equal parts (top and bottom) in Figure 2.

For calculations, a parallel version of FDS 4 was used. The calculations were done on an IBM SP2 computer using four processors. The total grid size was 360x250x20. This required about 10 GB of memory. The final computations were conducted for 360 seconds which took about 80 CPU hours.

4.3 Agent Model

Initially 48 agents were positioned in different rooms on the floor. Agents started moving towards the exits at the time fire started. Their path was described by a two-dimensional A* algorithm which is a well-known shortest path algorithm [8].

All unobstructed space was divided into nodes. The nodes were placed 14 cm apart. As agents move from one node to another, they select movements along the path to the nearest exit. Each evacuation path is then recalculated every second. The evacuation path could potentially change due to changes in fire conditions obtained from FDS 4 calculations. A certain value, or cost was assign to every node which represented the cost (F) of a move from one node to another. It was defined as

$$F=G+H+K\,, \tag{1}$$

G was a geometrical factor. If a move from one node to another node was horizontal or vertical, *G* was increased by 10. If it was diagonal, *G* was increased by 14. In addition the cost of a move was increased if the second node was on fire. That was determined by factor *H*. Factor *H* included effects of temperature, heat release, and smoke:

$$\mathrm{H} = \beta_1\,\mathrm{T} + \beta_2 \mathrm{HRR} + \beta_3 \mathrm{Soot}\ , \tag{2}$$

where *T* ,*HRR,* and *Soot* are maximum temperature (^{o}C), maximum heat release per unit volume (kW/m^3), and maximum soot density (mg/m^3) in vertical direction at the current agent position (*Xagent*, *Yagent*). $\beta_1, \beta_2, \beta_3$ are constants. Heat release information was used as a flame marker and visible position of the fire. The factor *K* estimated the distance between the position of an agent and the exit.

$$K = \left|Xagent - Xexit\right| + \left|Yagent - Yexit\right|\ , \tag{3}$$

where *Xagent* and *Yagent* are current x and y positions of the agent, *Xexit* and *Yexit* are x and y positions of the exit. From every position an agent could choose up to 8 possible directions to travel. The agent was limited by walls which considered impenetrable. The agents had collision-with-other-agents detection system as well. It prevented two or more agents to occupy the same node. A position one agent occupied at certain time was considered impenetrable by other agents at the same time. The agent moved to the lowest cost (F) position.

It can be seen from (1) - (3) that although an agent cannot walk through the wall it is capable of walking through fire. This consequence is supported by facts that in desperate situations people run through fire. Although it is possible for people to walk through fire it will take a toll on their health.

The agent health will decline while they are evacuating. The formula used in calculations takes into account the temperature, heat release, soot, carbon monoxide, and carbon dioxide. The agents start with health equal to one, decreasing according to the following formula:

$$Health_i - Health_{i+1} = \alpha_1 T + \alpha_2 HRR + \alpha_3 Soot + \alpha_4 CO + \alpha_5 CO2\,, \tag{4}$$

where $Health_i$ is health of an agent at time *i* second, *CO* is CO volume fraction (ppm), *CO2* is CO2 volume fraction (ppm). Soot density, CO and CO2 volume fractions are

maximum values in vertical (z) direction at the agent position as well as temperature and heat release. $\alpha_1,...,\alpha_5$ are constants. Agents died when their health reached a zero value. The formulas were based on real data for how these elements affect the human body. The CO data is shown in Table 1 [9].

Table 1. Effects of CO on agents

CO in the Atmosphere			% Blood COHb	Physiological Symptoms
Percent	mg/cubic meter	ppm		
0.007	80	70	10	Shortness of breath upon exertion
0.01	140	120	20	Shortness of breath upon moderate exertion, possible headache
0.02	250	220	30	Decided headache, dizziness, dimness of vision, impaired judgment
0.035-0.052	400-600	350-520	40-50	Headache, confusion, fainting, collapse
0.080-0.122	900-1400	800-1220	60-70	Unconsciousness, convulsions, death if continued exposure
0.195	2200	1950	80	Swiftly fatal

4.4 Agent-Fire Interaction

Current calculations were done under the assumption that agents do not affect the fire. They cannot either extinguish or increase intensity of the fire by their actions. On the other hand, the fire propagation affects both agent behavior and health. This assumption allowed us to have independent fire simulation. After finishing the fire simulation, the computed data was used as input for the agent simulations.

5 Results

Immediately after the ignition of kerosene and consumption of air in the room the fire moved from the room into the corridor and began propagating in the direction of air source, i.e. open windows and doors. Figure 3 shows temperature distribution at time 200 seconds, at 2.4 meters above the floor. At seen from Figure 3, 200 seconds after the fire started it has reached one of the building exits. The quickly-moving fire blocked some agents from exits.

Figure 4 shows evacuation visualization. The smoke shown represents the fire position. Agents are in the process of evacuating the building.

Agents far from the exits were more probable to die due to CO poisoning and smoke inhalation as a result of being blocked by the fire from exits. Through multiple runs of the scenario it was observed that disallowing escape through various exits

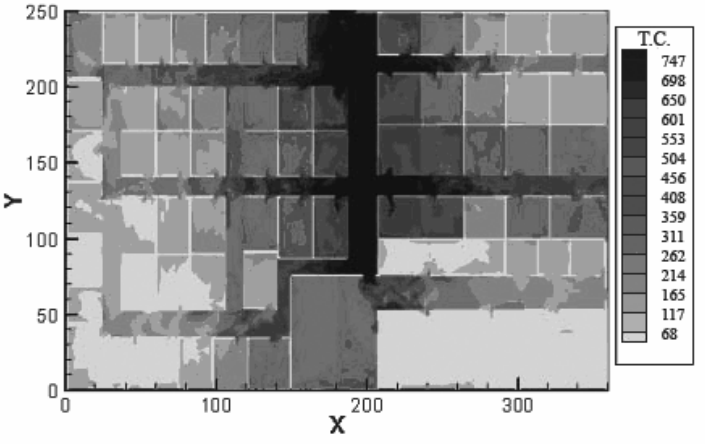


Fig. 3. Temperature distribution at time 200 sec. Vertical position is 2.4 m above the floor

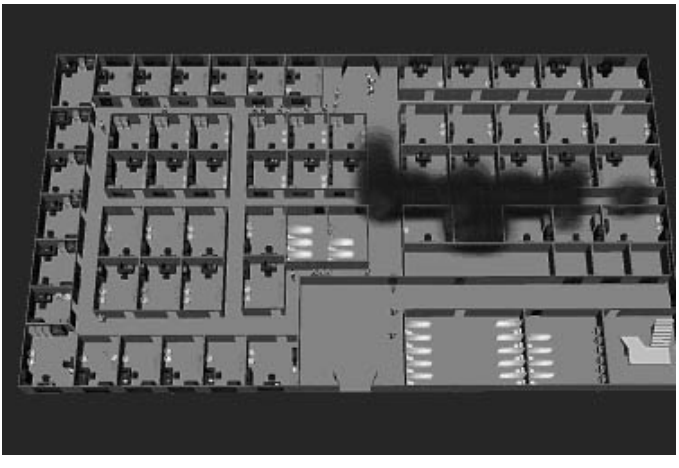


Fig. 4. Visualization of agent evacuation

dramatically affects agent death rate. The death rate with all exits open was 15 out of 48. In replicating the simulation with various combinations of exits closed during evacuation, it can be seen that the two main exits are critical to agent death rate. The death rate with just one of the main exits closed (the exit at the top of Figure 4) was 18 out of 48, a 6.25% increase in death rate by closing one exit. Closing both of the main exits (the top and bottom exits of Figure 4) has the most dramatic effect on the death rate: 28 out of 48 agents, a 58.3% death rate. This is a 20.8% increase compared to closing only one main exit. Figure 5 shows the evacuation paths of each agent as well as dead agent positions when all exits are open. Figure 6 shows the effect of closing the two main exits.

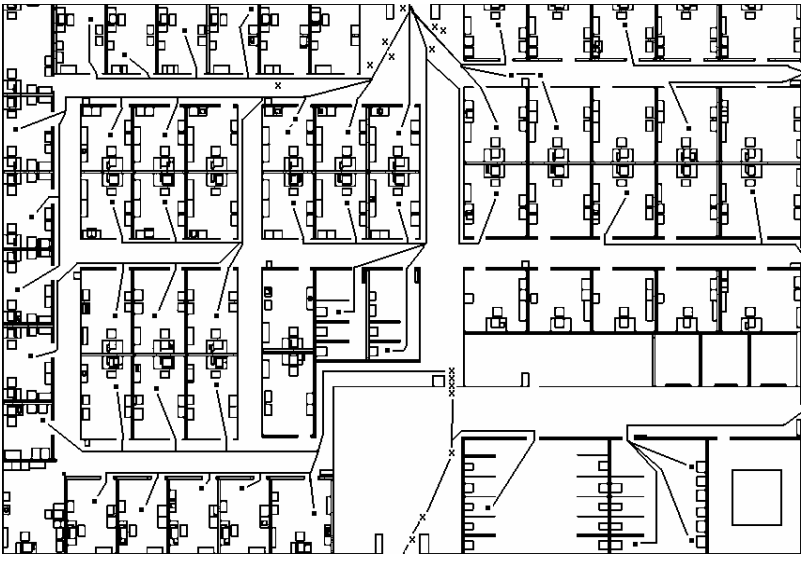


Fig. 5. Agent's evacuation paths with all exits open. Dots represent initial positions, X's represent dead agents

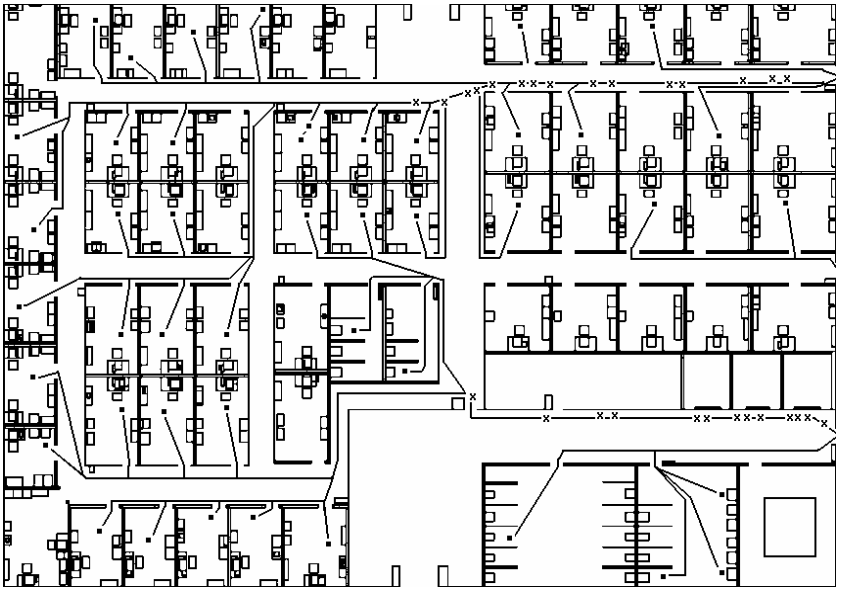


Fig. 6. Agent's evacuation paths with two main exits closed. Dots represent initial positions, X's represent dead agents

6 Conclusions

The state-of-the-art Shared Reality System was created and successfully tested in case of evacuation of a building on fire. The system included fire simulation (FDS), agent simulation, and the Shared Reality Engine for format conversion. The final results were presented in form of 3D movie visualization. Since fire and agent simulations did not interact directly with each other but through the shared reality, any one simulation can be replaced with a small change in the Shared Reality Engine to accommodate input/output stream from a new code. This change will not affect the remaining code. This provides a significant flexibility in selecting the most convenient and advanced software in each area separately.

References

1. Foong, C., Armstrong, B., Dilley, D., Grahn, J., Krull, K., Chaturvedi, A., Gore, J., Filatyev, S. "Towards Enabling A Distributed And Scalable Society Of Simulations." *2005 Spring Simulation Multiconference (SpringSim'05)* (2005).
2. Chaturvedi, A., Foong, C., Armstrong, B., Dilley, D., Grahn, J., Krull, K., Gore, J., Filatyev, S. "A Society of Simulations." *Submitted to Transactions on Systems, Man, and Cybernetics* (2005)
3. LS-DYNA. < http://www.ls-dyna.com/>
4. Popescu, V., Hoffmann, C., Kilic, S., Sozen, M., Meador, S. "Producing High-Quality Visualizations of Large-Scale Simulation." *Visualization, 2003. VIS 2003. IEEE* (2003)
5. McGrattan, K., Forney, G. "Fire Dynamics Simulator (Version 4). Users Guide." *NIST Special Publication 1019*. 2004 <http://fire.nist.gov/fds>
6. Xin, Y., Gore, J.P., McGrattan, K.B., Rehm, R.G., Baum, H.R. "Fire dynamics simulation of a turbulent buoyant flame using a mixture-fraction-based combustion model." *Accepted for publication in Combustion and Flame Journal* (2005)
7. Prasad, K., Baum, H. "Coupled Fire Dynamics and Thermal Response of Complex Building Structures." *Proceedings of the Combustion Institute* 30 (2004).
8. Rabin, S. (2002) *AI Game Programming Wisdom.* Charles River Media, MA.
9. Canadian Centre for Occupational Health and Safety, International Programme on Chemical Safety. 1999. Carbon Monoxide. <http://www.inchem.org/documents/ehc/ehc/ehc213.htm>

A Dynamic, Data-Driven, Decision Support System for Emergency Medical Services

Mark Gaynor[1], Margo Seltzer[2], Steve Moulton[3], and Jim Freedman[1]

[1] Boston University, School of Management, 595 Commonwealth Avenue, Boston
{mgaynor, jfreedma}@bu.edu
[2] Harvard University, Division of Engineering and Applied Sciences
margo@eecs.harvard.edu
[3] Boston University School of Medicine, Boston MA
smoulton@bu.edu

Abstract. In crisis, decisions must be made in human perceptual timeframes under pressure to respond to dynamic uncertain conditions. To be effective management must have access to real time environmental data in a form that can be immediately understood and acted upon. The emerging computing model of Dynamic Data-Driven Application Systems (DDDAS) fits well in crisis situations where rapid decision-making is essential. We explore the value of a DDDAS (iRevive) in support of emergency medical treatment decisions in response to a crisis. This complex multi-layered dynamic environment both feeds and responds to an ever-changing stream of real-time data that enables coordinated decision-making by heterogeneous personnel across a wide geography at the same time.

1 Introduction

Information technology has been associated with a shift in business strategy such as the shift in decision-making paradigms of the organization [1]. The emerging computing model of Dynamic Data-Driven Applications Systems (DDDAS) fits well in crisis situations where rapid decision-making [5] is essential. Effective decision-making in a crisis greatly depends upon immediate access and interpretation of local information within the context of the overall environment at any particular point in time. A key element of a crisis is its dynamic, turbulent, uncertain nature that creates both a sense of urgency and a need to reduce uncertainty in decision-making.

We view theories of network organizational decision-making [2][3] as the foundation for experimentation with emerging sensing networks and dynamic data-driven application technologies. The goal of these emerging technologies is to support decision-makers with relevant timely information that they can quickly interpret at the time and location that they must make a decision [4]. In this paper, we use emergency medical response to a mass casualty event as an illustrative example of how emerging technology supports a shift in decision-making during a crisis. In a

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 703 – 711, 2005.
© Springer-Verlag Berlin Heidelberg 2005

medical emergency situation, many split-second decisions are made in the process of locating, identifying, evaluating, treating and transporting a critically ill patient. When emergency medical personnel are dealing with multiple patients, all with various types of life or limb threatening injuries, critical decisions regarding who to treat, who not to treat, and in what order, must be made with split second accuracy to reduce the number of casualties as well as the number of patients with long term disabilities. Our emergency medical response application receives real time sensor and emergency resource operational data from the edge of the incident and presents this information to emergency medical personnel at the time and place that they need it to enable more effective patient management, including decisions that may effect human survival rates.

Mass casualty events (MCEs) are sudden, unexpected situations in which, over a short period of time, large numbers of casualties are generated and organized community support mechanisms are either crippled or destroyed [6]. A central problem in mass casualty events is the need to rapidly assess and respond to the situation by deploying multiple personnel and resources to monitor, triage, and track critical care patients. If the injured and severely injured cannot be triaged in a rapid and coordinated manner, their large numbers can quickly overwhelm emergency field personnel and hospital staff preventing them from providing quality trauma care [7]. A variety of problems may arise at large scale events, beginning with the inability to locate or gain access to patients in distress because of collapsed infrastructure, crowds, noise and other event specific circumstances (e.g., flooding or radiation). In addition, documenting care and monitoring patient status may become sporadic and disorganized due to multiple calls for assistance or many patients at one location within an event. All of these factors can lead to delays in recognizing when patients deteriorate, errors when administrating medications, and incomplete capture of patient information, including histories of the present illness, physical exam findings, and response to treatment. Furthermore, available patient treatment and outcome data may be lost during the triage and transfer process, as patients are handed from one team to another on their way to more definitive care.

In order to address the problems faced by EMT workers, we have developed iRevive, which is a mobile pre-hospital database system that allows point-of-care data capture in an electronic format. It consists of a network of wireless, handheld computers running the iRevive application, wireless patient location and vital sign sensors, a multi-frequency ambulance base station, and a central command center where field information is collected, viewed and stored. Wireless VitalDust sensors automatically capture and integrate patient location (GPS) and vital sign data (including heart rate and blood oxygen saturation) into each patient's pre-hospital record (Figure 1). This collated information is wirelessly and securely pushed over the Verizon 3G Wireless cellular network to a server that stores and relays selected patient information to a command center and hence to the receiving hospital.

Patient location and vital sign data is locally accessible to on-scene personnel to assist them with triage and decision-making processes. This same data is available in near real-time at the central command center, where a global view of the field

environment can be obtained and a greater semblance of order established. A properly trained triage officer at the central command center is better able to assist field personnel in determining the best course of action for each patient when that officer has immediate access to global information about the field environment and medical resources that are available at the trauma center, including bed and Operating Room availability, on-site staffing and transportation resources. Triage in the field and triage at the hospital gates therefore becomes an interactive, coordinated process that better matches the out-of-hospital caseload with critical, hospital-based trauma facilities and resources. More appropriate levels of medical care can be provided, thus limiting the number of casualties and reducing the number of patients with long-term disabilities.

Our emergency medical application is a Multi-Layer Dynamic Data-Driven Decision Support System. It provides decision support at three important layers: at the site of the incident, at local command centers, and at a central point of coordination. These three points form a hierarchical layering with on-site care at the bottom and central coordination at the top. At each layer, data is aggregated from layers below (similar to the different time variant hierarchical structure of Chaturvedi [5]). Each layer presents data aggregated from a dynamic set of real-time sensors and mobile Emergency Medical Technicians (EMTs). By linking real-time sensor data, procedural data, and geographic data to specific points in time, iRevive enables rapid decision support that considers actual EMT intervention and patient outcomes, as they occur to determine optimal future action.

2 iRevive Application Description

Figure 1 illustrates the overall system architecture of the triage/Geographic Positioning System (GPS) application (iRevive) [8][9]. It describes the current system under development (supported by NSF, NIH, and US army). This system has several major components including: Web Services to aggregate and present information, handheld mobile Personal Digital Assistants (PDAs) to provide situational awareness to each EMT, local command site for field coordination, central command site for global resource management, cellular/Satellite wireless link for real time communication between local and remote sites, infrastructure for real-time data transport between motes and local PDAs, and Vitaldust (Developed by Matt Walsh at Harvard University) patient tag (pulse/ox sensor, GPS receiver, micro-processor, data storage & transmitter) attached to patient.

On each patient, data from the GPS and medical sensors is combined on the Vitaldust patient tag and forwarded to the ambulance base station via a proprietary protocol. The local EMTs use their PDAs to view current vital-sign information and to enter patient medical information as treatment is performed in the field. These end-user devices also have business/medical rules, embedded in applications resident on the PDA, that help triage a group of patients based on their vital signs. Our end-user software provides several views of resources: a local view of patients, to assist a particular EMT, or a global view of all patients being treated by a group of EMTs,

which permits a commander to coordinate resources in the immediate area of an event. The overall system goal is to provide secure, end-to-end, real-time (including medical, environmental, and geolocation) information to first responders that allow situational awareness for local decision support and global management of resources. Aggregation of sensor data is made available to a centralized, or group of distributed command and control centers, via web services. Our emergency medical services application, iRevive, provides an electronic pre-hospital patient care record that combines manual entry of procedures provided in the field with real-time sensor data. This link enables valuable event data that can facilitate later mining to help determine better medical field procedures.

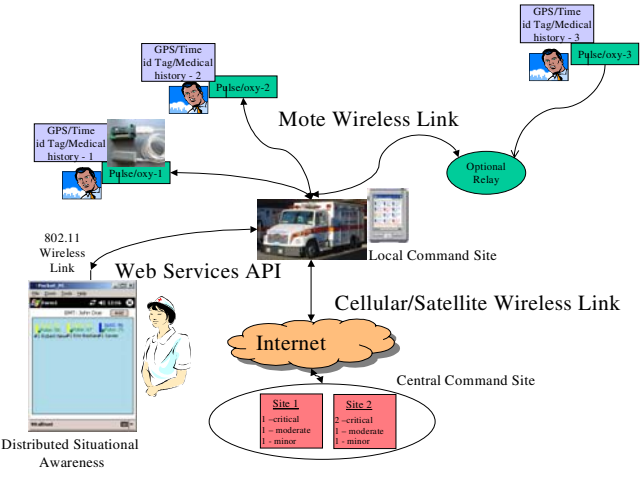


Fig. 1. iRevive System Architecture

The Vitaldust patient tag is based on the MICA mote, developed at UC Berkeley in collaboration with Intel Research. This device consists of an 8-bit Atmel ATMEGA 128L micro controller, 132K of memory, 512K of nonvolatile flash memory, and a 19.2 Kpbs radio operating in the 433 MHz or 916 MHz spectrum. These motes run the open source TinyOS operating system. The mote is interfaced to a pulse/oximeter and the Crossbow MTS420CA MICA2 Sensor board with a Leadtek 9546 GPS module. In addition to location and time, the MTS420CA also provides onboard dual-axis accelerometer, barometric pressure, humidity and temperature. The mote transports its data to a laptop computer in the onsite ambulance via a proprietary protocol (however, the industry is beginning to standardize around 802.15.4). These motes provide a powerful platform for experimentation with both digital and analog sensors.

Sensor data must be “application friendly” to facilitate widespread adoption of real-time sensor data within IT applications. Currently, the emerging web services standards are the dominant design for distributed exchange of data between applications. Our application has adopted these standards for both local and wide area

exchange of real-time sensor data. Local personnel utilizing applications on mobile devices have situational awareness with real-time access to sensor data via a web service. This local connectivity does not depend on Internet connectivity. When connected via a cellular or satellite link, the real-time sensor data is available, as a web service, to centralized or distributed command centers. Compliance with emerging open standards (such as web services) enables a flexible architecture in the context of exchanging data between heterogeneous systems in both the local and wide area.

The triage aspect of iRevive fits well into the paradigm of a DDDAS because its current view of the emergency response situation is based on a continuous flow of real-time, vital-sign sensor data combined with information input by the local EMT. Figure 2 illustrates the dynamic aspects of our triage application – the left side is the stable view, the right is a shifting view caused by changes in patients' conditions, which are based on changing vital signs. In future editions of iRevive, the medical rules that determine triage order will vary based on feedback from the EMT. Triage is a multi-layer application – each EMT has triage information based on the patients for whom they are responsible and each local commander has triage information for all local patients. The triage function aids local EMTs in their decision making process. Central coordination of global resources is critical to manage large-scale (or multiple) events that span large geographical areas.

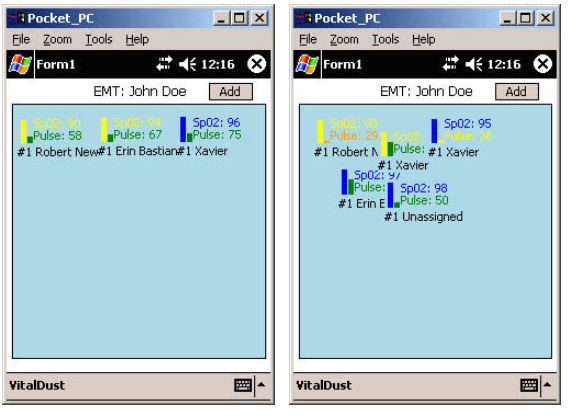


Fig. 2. Dynamic triage function in iRevive

2.1 Multi-layer Architecture

Figure 1 illustrates the three layers of decision-making that drive the iRevive decision support system: at the edge EMTs have situational awareness with sensor data and data from the local command center. The local command center receives both sensor data and procedural data from EMTS. At the highest layer the central command receives aggregated data from each local site and can also send real-time information to the local command centers.

The edge of an incident is where the EMTs and patients are physically located. Each EMT needs continuous access to real time sensor data of each patient to

determine the triage order of all patients assigned to them. This is accomplished by sending each EMT a continuous stream of vital sign data from each patient under the EMT's control. The primary concern in a crisis situation is that each EMT has situational awareness of their assigned patients based upon real time data. Information also flows from the local command center to each EMT, which might include data about a particular patient, treatment suggestions, or general instructions to the EMT. Providing real-time data to the EMT allows for situational awareness of their immediate time-critical responsibilities.

Data flows from each sensor network and is aggregated at the local command site along with data from each EMT. The local command center provides a view of all patients and EMTs at the local site allowing effective management of local resources. One primary application at this layer utilizes a PDA to receive data from the local command center to provide the EMT Commander a view of all local patients. Each local command center also receives data from the central command center for effective coordination of global resources. The local command site enables a view of all local resources combined with data from the central command center.

The top layer is the central command center that receives data aggregated from each local command site and includes real-time sensor as well as data input by each EMT about the care of each patient. This data enables resources to be managed across a broad region with many emergency events. The granularity of data required at the command center depends on the particular application. Systems that try to diagnose or suggest treatment algorithms need fine-grained patient-level data. However applications such as resource management across a distributed set of sites demand aggregated data. The overall architecture of iRevive enables data to flow between central and local command sites.

3 Limitations of Current Architecture

As described so far, iRevive relies on a centralized architecture, with a single ultimate destination for all data (the centralized control). While this is feasible in the short run, it is not a long-term, scalable solution. In reality, a major metropolitan area has a number of ambulance companies, multiple hospitals, and no central dispatching system that coordinates among them. Thus, the architecture does not reflect the reality of the particular application.

In a more general sense, the architecture as described here has fundamental limitations in its flexibility and scalability. In particular, there are a number of issues that it does not address: the ability to balance data flows across potentially overloaded network links, the ability to direct data to underutilized nodes, intermittent or disconnected operation in the presence of extremely high-bandwidth data streams, support for in-network data processing (e.g., aggregation, filtering, triggering, etc.), and optimizing the placement of in-network services to minimize overall bandwidth utilization

In the next section, we present a general architecture that addresses these limitations and provides a fundamentally new abstraction for the delivery of streaming, real-time data.

4 Hourglass

Hourglass is the networking infrastructure that we are developing to support the iRevive application environment. It is a particular application and implementation of a new abstraction called a Stream-based Overlay (SBON). Stream-based overlays are designed to provide the networking infrastructure to support dynamic, data-driven applications. Applications, similar to iRevive, are characterized by long-lived streams of real-time data that may demand high bandwidth. A network that is subject to multiple dynamic data-driven applications will easily become overloaded, so the infrastructure must be able to adapt to a rapid influx of data and reconfigure itself appropriately so as to avoid network overload, make efficient use of nodes in the network, and provide acceptable quality of service for the delivery of data from their source to their, potentially multiple, destinations.

The fundamental unit of dataflow in Hourglass and an SBON is the circuit. A circuit is a logical connection between one or more data producers and a data consumer, including zero or more in-network services. A service might be a filter (e.g., ignore all "normal" pulse readings), a trigger (e.g., perform some action whenever you see a pulse rate above 180 beats per minute), an aggregator (e.g., what is the average time from first contact to ambulance access), or an application-specific computation (e.g., match patients with particular emergency rooms). At any point in time, a common infrastructure may support many such circuits. The technical challenge in Hourglass is to realize a physical implementation of a logical circuit. This process entails locating the data producers and consumer, instantiating the appropriate services on intermediary nodes, and then setting up the particular data flows. As the data flow through these circuits, the system must also adapt to changes in network load, node load, and flow rate. This adaptation usually involves moving services around, instantiating new services, or denying admission to circuits that cannot adequately be supported.

Most of these technical challenges can be approached as an optimization problem. That is, we strive to minimize total network load subject to the response time constraints of a particular circuit description. More formally, a circuit is composed of some number of services that fall into one of the following categories: pinned, unpinned-unplaced, unpinned-placed. A pinned service is one that must reside on a particular node in the network. Data producers, such as sensors, are examples of pinned services; the system has no control over where these services can be located. An unpinned-unplaced service is one that can run anywhere in the network and that has not yet been assigned to a particular node. An unpinned-placed service is one that is free to run anywhere, but has already been assigned to a particular location. Our challenge then, is to place all the unpinned-unplaced services in a manner that minimizes network load subject to the application constraints. These constraints

might be response-time constraints or they might be social constraints (e.g., only a system under the control of an ER can assign a patient to a bed). Also, in the face of changing load conditions, the system may need to change the placement of unpinned services to continue meeting application constraints.

In our prototype Hourglass implementation, we have developed a decentralized network-aware service placement algorithm called Relaxation placement [10]. The main idea behind Relaxation placement is to partition the placement problem into two phases. First, an unpinned service in a circuit is placed in a virtual metric space (using latency) using a spring relaxation technique. Then, we map the virtual placement back into a real physical placement in the network. Performing data path optimization in our latency space has the advantage of naturally capturing global knowledge about bandwidth and latency in the network topology without imposing a large overhead due to network probing. Using our distributed spring-relaxation placement algorithm allows us to do this without requiring any centralized or global knowledge.

As load conditions change (either in the network, from the data source, or on particular nodes), our optimizer will move services as necessary. We build dampening into these optimization movements to avoid continual thrashing and reconfiguration.

While Hourglass is a solution specific to data sensor networks, these ideas and SBONs are applicable to the entire class of dynamic, data-driven applications, where consumers and producers enter and leave the system and bandwidth availability changes in unpredictable ways.

5 Conclusion

This paper described the iRevive EMT application in the context of a DDDAS. The multi-layer architecture illustrates the flexibility of the web services approach in sharing data among heterogeneous data sources. The feedback of real-time data from sensors, and other data sources provides multiple layers of decision support through iRevive. The limitations of the traditional web services model is described, and a new, more robust and scalable distributed data collection infrastructure called Hourglass is discussed.

Acknowledgments

NSF (ACI-0330244 and PFI-0227879), NIH (R41 RR018698-01A1), and US Army (Grant Contract No.:W81XWH-04-C-0113), and Matt Welsh and his research group.

References

1. Huber, G. P. (1990). "A Theory of the Effects of Advanced Information Technologies on Organizational Design, Intelligence, and Decision Making." *Academy of Management Review* **15**(1): 47-71.

2. Baldwin, C. and K. Clark (1997). "Managing in an Age of Modularity." *Harvard Business Review,* 75(5): 84-93.
3. Alberts, D. S. and R. E. Hayes (2003). Power to the Edge.
4. Strassner, M. and T. Schoch (2002). Today's Impact of Ubiquitous Computing on Business Processes. *International Conference on Pervasive Computing*.
5. Chaturvedi, A, Synthetic Environments for Continuous Experimentation – Working Paper
6. American College of Surgeons Committee on Trauma 1998. In: Resources for Optimal Care of the Injured Patient, 1999. Chicago, IL: American College of Surgeons, 87-91.
7. Frykberg ER, and Tepas JJ III, 1988. Terrorist bombings: lessons learned from Belfast to Beirut. Ann Surg 208:569 576.
8. Gaynor, M., and Welsh, M, and Moulton, S, Integrating Wireless Sensor Networks with the Grid, *IEEE Internet Computing*, special issue on the wireless grid, July/Aug 2004
9. Tollefsen, W, and Pepe, M, and Myung, D, and Gaynor, M., and Welsh, M, and Moulton, S, iRevive, a Pre-hospital Mobile Database for Emergency Medical Services, *International Journal of Healthcare Technology Management* (IJHTM) , Summer 2004
10. Shneidman, J., Pietzuch, P., Welsh, M., Seltzer, M., Roussopoulos, M., "A Cost-Space Approach to Distributed Query Optimization in Stream-Based Overlays," Accepted, *The Proceedings of the 1st IEEE International Workshop on Networking Meets Databases (NetDB'05)*, Tokyo, Japan, April 2005.

Dynamic Data Driven Coupling of Continuous and Discrete Methods for 3D Tracking*

Dimitris Metaxas and Gabriel Tsechpenakis

Center for Computational Biomedicine, Imaging and Modeling (CBIM),
Computer Science Dept., Rutgers University,
110 Frelinghuysen Rd, Piscataway, NJ 08854
{dnm, gabrielt}@cs.rutgers.edu

Abstract. We present a new framework for robust 3D tracking, using a dynamic data driven coupling of continuous and discrete methods to overcome their limitations. Our method uses primarily the continuous-based tracking which is replaced by the discrete one, to obtain model re-initializations when necessary. We use the error in the continuous tracking to learn off-line, based on SVMs, when the continuous-based tracking fails and switch between the two methods. We develop a novel discrete method for 3D shape configuration estimation, which utilizes both frame and multi-frame features, taking into account the most recent input frames, using a time-window. We therefore overcome the error accumulation over time, that most continuous methods suffer from and simultaneously reduce the discrete methods complexity and prevent possible multiple solutions in shape estimation. We demonstrate the power of our framework in complex hand tracking sequences with large rotations, articulations, lighting changes and occlusions.

1 Introduction

There are generally two major types of approaches to deformable and articulated shape and motion estimation: (i) the continuous ones that exploit the static and the temporal information in images, and (ii) the discrete ones that use only static information, i.e., they estimate the objects configuration based on a single frame. Continuous approaches are usually faster and more accurate than discrete approaches, but when they loose track they cannot easily recover due to error accumulation. On the other hand, discrete approaches can give a good approximation of an objects configuration without error accumulation over time. However, they have high computational cost and are based on searching in databases with limited number of object configurations.

In this paper, we introduce a new framework for robust 3D object tracking, to achieve high accuracy and robustness. Focusing on a specific case of tracking,

* This research has been funded by an NSF-ITR/NGS-0313134 and an NSF-ITR-[ASE+ECS]-0428231 Collaborative Project to the first author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 712–720, 2005.
© Springer-Verlag Berlin Heidelberg 2005

i.e., the $3D$ hand tracking, our approach is based on a *dynamic data driven* coupling of continuous and discrete methods; when our existing continuous tracking fails based on an error measure derived from the data, we can obtain efficient object configuration re-initialization using the discrete method presented in this work.

This paper is organized as follows. In the next subsection, we give a brief description of the previous work, including the existing continuous hand tracking method we used. In section 2 we describe the proposed discrete tracking scheme. In section 3 it is explained how the coupling between the two methods is achieved. In section 4 we present our results on the $3D$ hand tracking, including the case of sign language. Finally, section 5 describes our conclusions and our future work.

1.1 Previous Work

This paper focuses on hand articulations, where several techniques exist that treat the hand configuration estimation as a continuous $3D$ tracking problem [6, 18, 5, 8]. A possible drawback of some approaches is that they introduce additive errors over time, leading to the loss of the track, and when this occurs, they cannot usually recover. This is the reason why some discrete techniques have been developed in the last few years [3, 9], treating each frame independently from the previous ones, although they usually require higher computational time.

Both continuous and discrete methods for $3D$ hand tracking can be divided in two main classes: (a) the model-based ones [6, 12, 4, 12], where $3D$ hand models are constructed and a matching takes place between the input image features and the respective features of the model projection onto the image plane, and (b) the appearance-based approaches [10, 15], which involve mapping of the image feature space to the hand configuration space. Another problem that is tackled by some methods [15, 16, 3], is the background complexity, i.e. the discrimination between the hand and the background edges, when using edges as the visual cues for hand configuration estimation.

In the last few years, some approaches that use hand configuration databases have been proposed [13, 2] and the $3D$ hand pose estimation problem is converted into a database indexing one. The main problem that arises in these methods, apart from the computational complexity, is that multiple matches between the input hand image and the database samples may occur.

In the model-based continuous tracking of [6] that we use, $2D$ edge-driven forces, optical flow and shading are computed. They are converted into $3D$ ones using a perspective camera model, and the results are used to calculate velocity, acceleration and the new position of the hand. A Lagrangian second order dynamic hand model is used to predict finger motion between the previous and the current frame. A model shape refinement process is also used, based on the error from the cue constraints to improve the fitting of the $3D$ hand model onto the input data.

2 Discrete Tracking

For an input frame of the examined hand sequence, we extract 2D features, which will be used for describing the current frame, but will also be integrated with the respective features of a number of past frames to serve as multi-frame descriptors. Instead of matching between single images, we perform multi-frame matching between the most recent input frames and the samples from our synthetic hand database. As will be explained in section 3, the database search is efficient, when the discrete tracking is used in our integration scheme. We search our database in two steps: (i) according to general features, we find the most appropriate cluster, and (ii) using more detailed features we search for the best matching sample sequence inside the chosen cluster. The last hand configuration is chosen as the solution for the input frame. In this way, we avoid multiple matches, taking into account the most recently estimated hand configurations of the input video, without any additional computation load.

Hand Gestures Database. Our database contains configuration sequence samples, instead of single configurations as in [2, 13]. Our synthetic hand model has 20 $dofs$, as shown in Fig. 1 and its advantage is the good skin texture, that can be used for hand edge extraction.

We created 200 configuration sequences, under 29 views, and each sequence has $N_{max} = 15$ frames, which are enough to include tracking failures in the overall coupling scheme (for a $30fps$ input video). For each sample we have stored the N_{max} joint angle sets corresponding to its successive configurations. We have also extracted and stored (i) the single frame and (ii) the multi-frame descriptors of each configuration sequence, as described below.

The database is organized according to which side of the hand is visible (projection information) in the last frame, and how many fingers are visible in the first and last frame of each sample sequence. Thus, we have divided our database into 108 clusters, each one containing 54 samples on average.

2D Hand Features. For every input frame, we use as descriptors both boundary and region-based information of the captured hand.

Single Frame Features. In order to estimate the $3D$ hand configuration by searching in our database for the best matching configuration sequence, we use the following cues. *(i) Boundary-based features:* For each input frame we ex-

Fig. 1. Our virtual hand and the 20 $dofs$ describing all possible articulations

tract the contour of the hand and the corresponding curvature function and Curvature Scale Space (CSS) map [7, 1]. The CSS peaks indicate the most important zero-crossings Z_c of the contour. In this way curvature is an efficient shape descriptor for shape changes under complex movements and scaling. *(ii) Region-based features:* We extract the edge map of each input frame, using the canny edge detector, and we calculate the edge orientation histogram of the hand, as in [15, 3], with $B = 45$ bins. The orientation histogram can provide us with information about the edges in the interior of the hand. *(iii) Projection information:* From the currently estimated configuration, we obtain the pose information for the next frame, i.e. which side of the hand is visible (palm, side or knobs view), assuming that the hands general pose does not change significantly in two successive frames. *(iv) Finger counting:* For each input frame we count the clearly visible fingers F, by calculating the most important zero-crossings extracted in (i): $F = \frac{Z_c}{2}$.

Multi-frame Features. Instead of searching and matching hand configurations in the database, we search for configuration sequences, by taking into account the N_{max} most recent frames of the input sequence. To reduce the computational complexity, we integrate the extracted $2D$ single frame features into two vectors. For P points of an object contour and its curvature (B bins of the edge orientation histogram), over N_{max} successive frames of the input video segment, we can assume that we have P points (B points) in an N_{max}-dimensional space, while we need to have P points (B points) in the $1D$ space. Thus, the problem is transformed into a dimensionality reduction task. For the hand tracking application, we used the nonlinear local Isomap embeddings proposed by Tenenbaum et. al. [14], keeping $(P, B) << N$. We chose to use Isomap, instead of using a linear embedding e.g. PCA (Principal Component Analysis), because we have nonlinear degrees of freedom, and we are interested in a global *hand movement signature*, i.e. a globally optimal solution [11].

Thus, if $\overline{K} = [K_n | n = 1, \ldots, N_{max}]$ and $\overline{H}_\vartheta = [H_{\vartheta,n} | n = 1, \ldots, N_{max}]$ are the sets of N_{max} curvature functions K_n and edge orientation histograms $H_{\vartheta,n}$, extracted over N_{max} frames, the embedded $1D$ multi-frame descriptors are respectively,

$$\tilde{k} = M^{(N,1)}(\overline{K}), \;\; and \;\; \tilde{h_\vartheta} = M^{(N,1)}(\overline{H}_\vartheta), \tag{1}$$

where $M^{(N,1)}$ represents the Isomap embedding from the N to the 1 dimensional space. Fig. 2 illustrates four examples of gesture signature using the embedded CSS descriptor. Each row represents one of the four illustrated hand movement cases, whereas the first column shows the respective embedded CSS descriptors.

Matching Between Input Frames and Database Samples. The matching criterion is the undirected chamfer distance. In general, given two point sets A and B, their undirected chamfer distance $d(A, B)$ is defined by the forward $d^f(A, B)$ and backward $d^b(A, B)$ distances:

$$d^f(A,B) = \frac{1}{\|A\|} \cdot \sum_{a^i \in A} \min_{b^j \in B} \|a^i - b^j\|, \;\; d^b(A,B) = \frac{1}{\|B\|} \cdot \sum_{b^i \in B} \min_{a^j \in A} \|a^j - b^i\|, \quad (2)$$

$$d(A,B) = d^f(A,B) + d^b(A,B) \quad (3)$$

Replacing A and B with the multi-frame descriptors $\tilde{k} = [\tilde{k}^i | i = 1, \ldots, P]$, $\tilde{k}_s = [\tilde{k}_s^i | i = 1, \ldots, P]$, we obtain the chamfer distance d_k between the embedded curvature functions of the input video segment and the database configuration sequence, respectively. Similarly, replacing A and B with $\tilde{h}_\theta = [\tilde{h}_\theta^i | i = 1, \ldots, B]$, $\tilde{h}_{\theta,s} = [\tilde{h}_{\theta,s}^i | i = 1, \ldots, B]$, we obtain the respective chamfer distance d_h between the embedded edge orientation histograms. In Fig. 2 we present four gestures (four key-frames are shown), where the first one ((b1)-(e1)) is completely different from the other three, the second one ((b2)-(e2)) is similar to the last two, which are the same gesture performed twice. The undirected chamfer distances between the corresponding embedded curvature functions of Fig. 2(a1)-(a4) are $d_k^{(a_1),(a_2)} = 10.33$, $d_k^{(a_1),(a_3)} = 7.87$, $d_k^{(a_1),(a_4)} = 8.54$, $d_k^{(a_2),(a_3)} = 6.93$, $d_k^{(a_2),(a_4)} = 6.31$, and $d_k^{(a_3),(a_4)} = 1.21$, where superscripts indicate the corresponding cases compared. It can be seen also numerically that cases (a3) and (a4) of Fig. 2 are the most similar.

For a given database cluster $S \in \mathbf{S}$, where $\mathbf{S}$ is the set of all clusters, i.e. the entire database, and a set of input frames u, the best matching sample $\hat{s} \in S$ $(\in \mathbf{S})$is given by,

$$\hat{s} = \arg\max_{s \in S} p(s|d_h(u,s)) \cdot p(s|d_k(u,s)), \quad (4)$$

where $d_h(u,s)$ and $d_k(u,s)$ are the extracted chamfer distances. The two matching probability functions $p(s|d_h(u,s))$ and $p(s|d_k(u,s))$ are gaussian distributions.

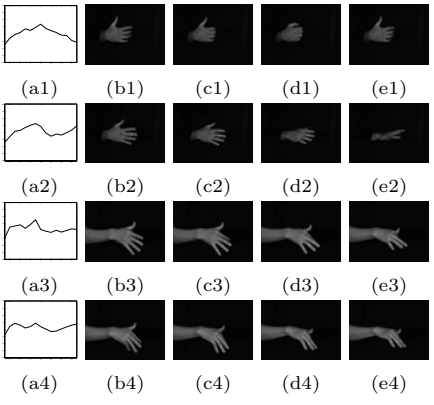


Fig. 2. Embedded CSS descriptors: (a1)-(a4) are the $1D$ embedded CSS, and (b1)-(e1), (b2)-(e2), (b2)-(e2) and (b2)-(e2) are key-frames of the respective sequences

3 Dynamic Data Driven Coupling of Continuous with Discrete Tracking

Our approach is based primarily on the continuous tracking method of [6], briefly described in 1.1. During each tracked frame we dynamically search in the synthetic hand database and choose the cluster that best matches the corresponding descriptors of our input sequence segment. In particular, our cluster search is based on the pose of the hand in the last (current) frame and the number of fingers visible in the first, and the last frame of the video segment. In this way, for each time step we know which database cluster best matches with the N_{max} most current frames of the input sequence. When there is an indication that the continuous method is about to loose track, we use the discrete approach, where we search in the best matching cluster, for the best matching sample sequence, using the multi-frame descriptors.

For each frame of the input video, the difference between the model projection onto the image plane, and the tracked hand gives us the indication whether continuous tracking fails or not. In order to estimate this difference, we use the undirected chamfer distance d_{map} as in eqs. (2),(3), between the model and the examined and edges. Thus, what we need to learn is the joint probability of continuous tracking given the distance d_{map} between the hand and the estimated model: $p(\mathbf{c}|d_{map})$.

We applied continuous tracking off-line, including cases where it fails, and we concluded that this happens when there are complex hand movements, such as strong articulations and rotations, i.e. where optical flow estimation fails. We performed complicated movements and articulations between simple hand movements (translations or rotations parallel to the camera plane), i.e. cases where it is successively $\mathbf{c} = \{1, 0, 1\}$ (continuous tracking success-failure-success). We used five hand sequences, $3,600$ frames each, where a hand performed the same pattern ($\mathbf{c} = \{1, 0, 1\}$) under many different articulations. The joint probability $p(\mathbf{c}|d_{map})$ was then learnt using a a linear two-class SVM [17].

In summary our method is based on the use of two different types of tracking methods. At each time only one is used based on an error measured defined by the input data. This approach results in significantly more robust tracking results as described below.

4 Experimental Results

After learning off-line the probability $p(\mathbf{c}|d_{map})$, we automatically estimated the threshold for d_{map}, under which the continuous tracking can be applied safely, as $T \simeq 0.27$.

In Fig. 3 the upper images of each row illustrate the case of a strong rotation, where fingers get occluded, while the lower images of each row show the extracted final result based on coupling between continuous and discrete tracking. In this case, the hand performs a strong rotation without any significant

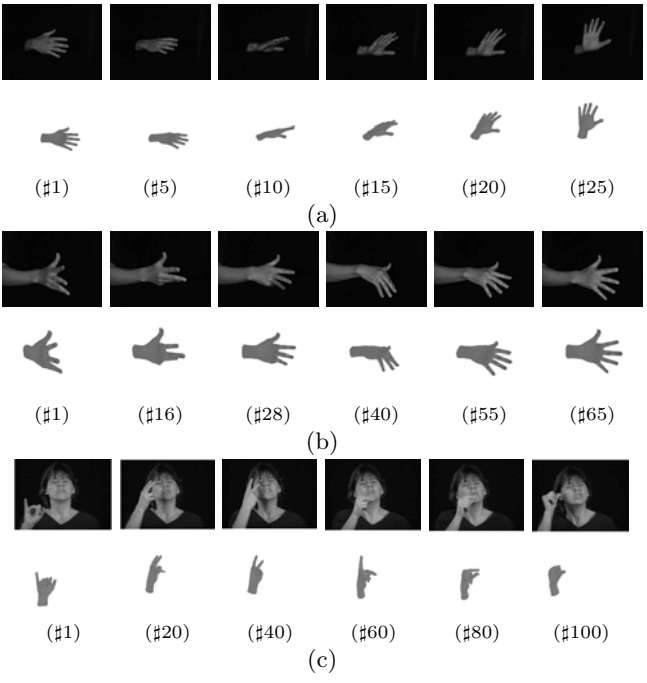


Fig. 3. Tracking results using our coupling framework

finger articulations and continuous tracking is performed in long time intervals without any significant error.

Fig. 3(b) shows a more difficult case, where the hand performs not only complicated rotations but also finger articulations. In this case, the time intervals where continuous tracking was successfully applied are smaller, i.e. in the coupling scheme discrete method initializes the hand model in more frames.

In Fig. 3(c), we illustrate the case of tracking a hand performing sign language, including both fingerspelled letters and continuous signs. Most of the fingerspelled letters of sign language cannot be captured with continuous tracking and that is why its coupling with a discrete scheme is crucial. The illustrated key-frames are taken from the phrase 'Who did John see yesterday?', which in sign language is said as 'John (fingerspelled) - see - who - yesterday (?)'. Our coupling method performs well even in this case, where there are lighting chances and complicated background, which are handled by the continuous tracking method of [6].

5 Summary and Conclusions

We have presented a Dynamic Data Driven Application System (DDDAS) for 3D object tracking in monocular sequences based on a novel coupling of continuous and discrete tracking methods. We have focused on 3D hand tracking, since it is

a very challenging task with a wide variety of applications. We have shown how our approach handles complex articulations, abrupt and large movements and occlusions. Our aim is to further evolve our method to be used for tracking of a much larger type of articulated and nonrigid motions. In particular we plan to further analyze signed languages such as ASL and also use our method in HCI applications.

References

1. S. Abbasi, F. Mokhtarian and J. Kittler, "Curvature Scale Space Image in Shape Similarity Retrieval," *Multimedia Systems*, 7(6), pp. 467-476, 1999.
2. V. Athitsos and S. Sclaroff, "Database Indexing Methods for 3D Hand Pose Estimation ," *Gesture Workshop*, Genova, Italy, April 2003.
3. V. Athitsos and S. Sclaroff, "Estimating 3D Hand Pose from a Cluttered Image," *IEEE Conference on Computer Vision and Pattern Recognition*, Wisconsin, June, 2003.
4. J. Lee and T. Kunii, "Model-based Analysis of Hand Posture," *IEEE Computer Graphics and Applications*, 15, pp. 77-86, 1995.
5. J. Lin, Y. Wu and T.S. Huang, "Modeling the Constraints of Human Hand Motion," *5th Annual Federated Laboratory Symposium (ARL2001)*, Maryland, 2001.
6. S. Lu, D. Metaxas, D. Samaras and J. Oliensis, "Using Multiple Cues for Hand Tracking and Model Refinement," *IEEE Conference on Computer Vision and Pattern Recognition*, Wisconsin, June 2003.
7. F. Mokhtarian and A. Mackworth, "A Theory of Multiscale, Curvature-based Shape Representation for Planar Curves," *Pattern Analysis and Machine Intelligence*, 14(8), pp. 789-805, 1992.
8. J. Rehg and T. Kanade, "Model-based Tracking of Self-occluding Articulated Objects," *IEEE International Conference on Computer Vision*, Cambridge, MA, June, 1995.
9. R. Rosales, V. Athitsos, L. Sigal and S. Sclaroff, "3D Hand Pose Reconstruction Using Specialized Mappings," *IEEE International Conference on Computer Vision*, Vancouver, Canada, July, 2001.
10. N. Shimada, K. Kimura and Y. Shirai, "Real-time 3D Hand Posture Estimation based on 2D Appearance Retrieval Using Monocular Camera," *IEEE ICCV Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems*, Vancouver, Canada, July 2001.
11. V. de Silva and J.B. Tenenbaum, "Global versus Local Methods in Nonlinear Dimensionality Reduction," *Advances in Neural Information Processing Systems 15*, (eds.) M.S. Baker, S. Thrun and K. Obermayer, Cambridge, MIT Press, pp. 705-712, 2002.
12. B. Stenger, P.R.S. Mendonca and R. Cipolla, "Model-based 3D Tracking of an Articulated Hand," *IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, December 2001.
13. B. Stenger, A. Thayananthan, P.H.S. Torr and R. Cipolla, "Hand Pose Estimation Using Hierarchical Detection," *International Workshop on Human-Computer Interaction*, Prague, Czech Republic, May 2004.
14. J.B. Tenenbaum, V. de Silva and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science Magazine*, 290, pp. 2319-2323, 2000.

15. A. Thayananthan, B. Stenger, P. H. S. Torr and R. Cipolla, "Shape Context and Chamfer Matching in Cluttered Scenes," *IEEE Conference on Computer Vision and Pattern Recognition*, Madison, June 2003.
16. J. Triesch and C. von der Malsburg, "A System for Person-Independent Hand Posture Recognition against Complex Backgrounds," *Pattern Analysis and Machine Intelligence*, 23(12), December 1999.
17. V. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.
18. H. Zhou and T.S. Huang, "Tracking Articulated Hand Motion with Eigen Dynamics Analysis," *IEEE International Conference on Computer Vision*, Nice, France, October 2003.

Semi-automated Simulation Transformation for DDDAS

David Brogan, Paul Reynolds, Robert Bartholet, Joseph Carnahan, and Yannick Loitière

Computer Science Department,
University of Virginia, Charlottesville, VA 22901, USA
{dbrogan, pfr, bartholet, carnahan, ycl2r}@virginia.edu

Abstract. Ultimate DDDAS success demands that DDDAS simulations be increasingly reconfigurable and adaptable to a growing variety of runtime sensor feedback. Because we expect a simulation's requirements to change during its lifetime, a new emphasis is placed on designing simulations that are prepared for transformation. In this paper, we address this new interest in designing for transformation. Our technology combines the specialized insight of simulation designers with the principled application of automation techniques to capitalize on untapped human and computational resources. In addressing simulation transformation issues that arise at design time, composition time, and runtime, we demonstrate how a semi-automated process impacts the entire simulation life cycle. The resulting suite of simulation transformation tools supports the crosscutting needs of DDDAS practitioners.

1 Introduction

Dynamic, data-driven application systems (DDDAS) pose challenges that reflect the core of simulation technology's hardest problems: composition, design for reuse, multiresolution modeling, interoperability, and scenario-driven adaptation. DDDAS's complexity, compounded by the uncertainty with which data and resources will be available, requires adaptability - a capability to adjust to match runtime conditions. Many DDDAS research efforts have produced adaptability results suitable for the application under study, and some span even a wider set of applications. We are investigating critical crosscutting adaptability issues that are independent of any particular application. We report on the COERCE effort now underway at the University of Virginia, where our primary focus is a semi-automated transformation process intended to support dynamic simulation modification and refinement to match runtime conditions.

We are exploring simulation transformation because the anticipation of and provision for all specific uses of a simulation, over its lifetime, is generally unattainable. A better method for addressing the uncertainty inherent in DDDAS is to identify effective means to control a simulation and to explore and model the impact of those controls. Because there are so many things that can impact simulation behavior, a designer must identify the specific attributes that permit the

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 721–728, 2005.
© Springer-Verlag Berlin Heidelberg 2005

greatest exploration of "structural uncertainty" [1] using semi-automated methods. These explorations can be cataloged using machine learning and utilized at runtime to adapt to unexpected conditions. Because we anticipate DDDAS applications will routinely combine multiple simulation components to achieve a larger DDDAS objective, we are also investigating the design of components with transformation in mind.

We first describe a method for capturing designer knowledge in language constructs at the outset of a simulation's design in order to enable the creation of logically tractable and easily transformed instances of a simulation. We differentiate between capturing the nature of unexpected conditions, which we believe is possible, and the specifics of those conditions, which experience has shown to be difficult to impossible. We argue that capturing the nature of abstract conditions is useful in many ways: 1) Later users can knowledgeably review applicability of a simulation to a specific set of DDDAS conditions, 2) If new conditions merit simulation transformation, those involved will have the best information available to support the effort, and 3) If future transformation is possible, DDDAS simulations can be instantiated to accommodate narrow conditions, thus increasing expectations for cost savings.

Next we present our investigation of "applied component synthesis" - our approach to component-based simulation that keeps the practitioner's objectives in mind. We specifically consider the synthesis of transformable components. All of the component synthesis cost analyses we have seen to date do not take the potential for transformation into account when considering the cost of synthesis. We do. We present our work in the context of contemporary composition and software reuse results from the software engineering and simulation communities. Our analysis of contemporary results identifies the sources of complexity when dynamically composing or reusing simulations and motivates the need for adaptable simulations.

Lastly, we describe a simulation transformation tool that automatically steers a simulation towards desired outputs. These desired outputs typically come from external observations and they serve as corrective feedback for simulations. Without anticipation of such observations, the simulation will not have a preconfigured feedback loop to meet the desired outputs. We demonstrate that offline experimentation can produce an abstraction of simulation behavior that serves to chart a course from the runtime simulation's current output to a desired output.

Our results align well with the four classes of DDDAS research challenges identified by Darema [2]: applications, mathematical algorithms, systems software, and measurements. Our presentation of ways designer insight can be captured and expressed through programming language primitives provides a "multi-modal method for describing the system at hand" in order to handle dynamically injected data. We address the needs of applications to be dynamically composed from families of models according to streamed data and our automatic simulation transformation tool permits well-behaved, runtime steering of a simulation to accommodate external observations.

2 Building Transformable Simulations

Modular and adaptable software has been the center of software engineering research for decades. However, software reuse remains time consuming and difficult to automate outside of specific application domains [3, 4, 5]. We view simulation as another potentially fruitful domain. We posit that designers possess critical knowledge about a simulation's flexibility and limitations at design time, even if runtime circumstances are unpredictable. We focus on methods for capturing this knowledge so transformation can be facilitated. We place special emphasis on capturing insights about time-management and event-generation algorithms –the most commonplace and important of a simulation's attributes [6]. This focus distinguishes our work from mainstream software reuse research.

2.1 Capturing Designer Insight

Simulations have characteristic elements that lend themselves to automated exploration and manipulation, which we call "flexible points" [7]. Many of these flexible points reflect the design decisions that were made to construct the simulation, such as the use of stochastic sampling, the selection of input distributions, and the level of detail included in the model. In order to facilitate the process of simulation transformation, we are developing categories of flexible points [8]:

- Decisions made during conceptual design of a simulation
- Decisions made regarding low-level implementation details
- Opportunities to select a variable or function assignment from a set
- Opportunities to select a variable or function assignment from an ordered set (for automatic evaluation using local search techniques)
- Opportunities to reorder events
- Opportunities to adjust level of abstraction or remove aspects

Useful information to record for each flexible point includes how its value could be changed (by changing a parameter or by executing an alternative section of code) and what effects the change will have on the simulation's behavior.

Different types of flexible points and different types of information that must be captured define a two-dimensional space. We use language constructs to encode the flexible point descriptions in the simulation code itself. Metalanguages such as XML, for example, can then parse these constructs and use existing libraries and tools to generate, display, and manipulate the flexible points. This simplifies the development of simulation prototypes and the evaluation of the usefulness of different types of flexible points for DDDAS.

2.2 Temporal/Event Manipulation

An attribute that distinguishes simulations from general-purpose software is the role of time and events in dictating the program's semantics. It is possible to represent a simulation as a (possibly repeating) timeline of events or intervals, reflecting the way the simulated phenomenon itself occurs in time. We are exploring possible

sets of operations that can be performed on timelines, such as looping, splicing, and overlaying timelines. From another perspective, simulations can be viewed as sequences of event generation actions (as opposed to event graphs, as captured in [9]). We are developing a transformation language for "event-generating systems" wherein the nature of an event generation may not be fully determined until the time when it is created by a previous event in the simulation.

One of the current difficulties with transformation of DDDAS simulations is the risk that the data may call for an unexpected series of changes, pushing the simulation out of the domain in which it was originally validated. However, by developing a set of well-specified operations on timelines or event systems, it becomes feasible to analyze a set of transformations on a simulation in advance. Without knowing exactly which operations will be applied or in which order, we can still explore whether or not the validity of the simulation is preserved under each of the operations in the transformation language. Such explorations could be conducted using model checking, e.g. [5] , for example. Then it becomes possible to dynamically apply analyzed transformations in response to external observations without risking the stability of the simulation system.

3 Composability

A crosscutting research challenge in DDDAS is the capability to dynamically select and compose models based on runtime conditions [2]. We are investigating model composition. Past research indicates that in the general case component selection is inherently intractable, but we believe there are opportunities to relax typical assumptions in order to reduce the problem to one that is practical in applied modeling systems. We have discovered a relaxed form of composition that is practical, in the context of an efficient means of transformation. We have labeled our approach "applied component synthesis."

3.1 Software Composition

In the software engineering community, composability is typically discussed within the framework of component-based software design (CBSD). Primary CSBD models include Microsoft's Component Object Model Plus (COM+), the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA), and Sun's Enterprise JavaBeans (EJB). These technologies are similar in that they enforce a binary structure for exposing public interfaces allowing components to provide services to clients, which may or may not themselves be components. While CBSD technologies provide the facilities to communicate and provide services, they make no guarantees about the meaning, reliability, or consistency of the exchanged information.

In his widely read text on component software, Szyperski defines a composition as an "assembly of parts (components) into a whole (a composite) without modifying the parts" [10], which is consistent with the software engineering view of composition as functional composition – composed black boxes with clearly de-

fined interfaces. Common exemplars are graphical user interfaces (GUIs), mathematical libraries, and UNIX tools. Functional composition is very restrictive for the modeling domain where practitioners typically want to view components as white boxes, and to reason about the semantics of the internals. We are investigating the means to reason about the semantics of models so formal methods of simulation composition can ensure semantic compatibility is preserved [6].

3.2 Simulation Composition

In the simulation community, composability has been defined as "the capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements" [11]. In contrast to Szyperski's definition, this definition admits reasonable amounts of model transformation. Within the simulation community both composability and interoperability have been topics of intense study. Composability prohibits the use of substantial integration efforts to combine components to meet requirements. Interoperability, on the other hand, permits a one-time integration effort as is commonly used in Distributed Interactive Simulation (DIS), Aggregate Level Simulation Protocol (ALSP), and the High Level Architecture (HLA). These technologies provide practical tools for interoperability, but with the same drawbacks seen in current CBSD technologies.

Recent interest in the M&S community has been focused on the simulation composability component selection problem [12, 13, 14]. Informally, the component selection problem seeks to find the subset of preexisting components that meet a simulation's objectives (requirements). The component selection problem does not address changing objectives, rather it assumes that all objectives are known at selection time. We believe a more flexible approach, recognizing the reality of changing requirements, is needed. Furthermore, component selection, as currently framed, assumes that components, to be composable, must be both syntactically and semantically composable. We agree with Page et al. that the easier problem is syntactic composability [15] and composability efforts should focus on the hard problem of semantic composability.

Only one theoretical model of simulation composability has been proposed to date [16, 17]. The authors define models as functions, and simulations as the process of stepping through the functions, with each step allowing the model to receive input, give output, and change state. Although this model addresses functional composability and semantic correctness, we find it too limited by traditional functional composition assumptions (e.g. no commutativity).

3.3 Breaking the Composability Paradigm

We reject the common assumption of immutable components. We do not agree with the assumption that a master set of components exists that can satisfy all possible requirements. In the domain of computational sciences, we believe practitioners, too, will not permit themselves to be limited by such an assumption. A more practical solution permits transforming existing components when a

composition is sufficiently close to satisfying all requirements. Component transformation is a particularly compelling technology when considering the challenge of adapting to requirements that change at runtime.

We are currently investigating a new paradigm of simulation reuse, applied component synthesis, that relies on efficient transformation. Applied component synthesis requires a formal model describing an iterative process of selecting, transforming, creating, validating, and assembling simulation components to meet changing requirements. Initial work regarding component selection algorithms is underway as is the investigation of metrics and means that will guide practitioners in component synthesis. We envision developers making decisions on the importance of different aspects of time and resources, encoding these decisions, and then allowing this encoding to facilitate the process of component selection and transformation.

Our initial results include a new model for the component selection problem – a model that permits consideration of component transformation in a component selection cost analysis. We have demonstrated that cases of this problem are NP hard, and others are polynomial time. We are utilizing our results to identify heuristics and to conduct further reconsiderations of assumptions.

4 Automatic Behavior Modeling

Incorporating data into an executing simulation has the potential to improve the simulation's utility by identifying and correcting errors, steering the simulation towards more effective states, or triggering the reconfiguration of the simulation's underlying computational components. In each of these cases, the infused data may trigger large corrections in the simulation state, which can have undesirable side effects when internal simulation variables are not changed accordingly. Such instabilities should be expected in complex simulation systems because they are composed of a dynamic set of interacting software components, each possessing internal variables and non-linear equations. Because state changes of large magnitude are common sources of such instabilities, we propose a method where a desired change in state is subdivided into multiple smaller changes that avoid simulation discontinuities and perturbations.

Consider how runtime data may reveal that the output O of a simulation in state S is not tracking actual conditions O' of the world and a new simulation state S' will be required to realign the simulation. The simulation (S, O) must be transformed to (S', O'). Recalling that a simulation's variables cannot directly be modified without causing unforeseen complications, we seek an iterative transformation process that permits the simulation to update all state and internal variables synchronously. The challenge becomes one of finding a sequence of states $S_1, S_2, ..., S_n$ that incrementally takes the simulation output from O to O' while avoiding the problems caused by the internal variables. We require a model that provides a means to select the intermediate steps between O and O' and permits us to map them to the corresponding $S_1, ..., S_n$ that generate them. To build this model, we use a data-driven approach that uses

simulation observations acquired during typical execution of the simulation. A particular feature of this model will be to constrain interpolation to outputs that have been observed so that the sequence between O and O' will be supported by valid simulation states.

We use a modeling tool called self-organizing maps to accomplish our simulation transformation goals. Self-organizing maps are a family of neural network-based clustering algorithms developed by Teuvo Kohonen in the 80s [18]. The self-organizing map uses a network of neural network nodes to record simulation outputs observed during a training period and it generates a similarity relationship such that similar outputs get placed in neighboring locations on the map. We benefit from this in that we can use locally guided path finding algorithms to search for an acceptable path from O to O'. While training the self-organizing map, each map node is associated with the simulation outputs that it best matches and their corresponding simulation states. From this mapping, a path through the self-organizing map's output space can be used to generate the desired $S_1, ..., S_n$.

To test this data-driven simulation transformation process, we implemented a curve-plotting program that generates non-linear and counterintuitive relationships between the simulation state and its output. The simulation state is a vector of five continuous parameters and the output is a 64x64-pixel image of a parametric curve. A fully connected 10x10 self-organizing map was trained on a set of 1,000 output images generated by a parameter sweep through four of the five state parameters. Associated with each output image placed in the map is the image's corresponding parameter setting. Given a pair of source and target images, we can then use the map to generate a path (using any path finding algorithm) of similar images between them. In ongoing work, the corresponding sequence of simulation states will be generated to smoothly transition from the source simulation state to a new state satisfying the output criteria.

5 Conclusion

In this paper, we have presented three research projects underway at the University of Virginia in support of the COERCE effort. These projects are united in their combined use of simulation designers and automation to address transformation challenges that would otherwise be insurmountable. Each simulation transformation project addresses a different point in the simulation life cycle and addresses the crosscutting needs of the DDDAS community. We will continue to pursue ways a simulation designer's insight and automation can be brought to bear on transformation's challenges.

Acknowledgments. The authors gratefully acknowledge the support of the NSF under grant 0426971.

References

1. Davis, P.: Exploratory analysis enabled by multiresolution multiperspective modeling. In: Proceedings of the Winter Simulation Conference. (2000)
2. Darema, F.: Dynamic data driven application systems: A new paradigm for application simulations and measurements. In: Proceedings of the International Conference on Computational Science. (2004)
3. Garlan, D., Allen, R., Ockerbloom, J.: Architectural mismatch or why it's hard to build systems out of existing parts. In: Proceedings of the International conference on Software engineering. (1995) 179–185
4. Sullivan, K.J., Knight, J.C.: Experience assessing an architectural approach to large-scale systematic reuse. In: Proceedings of the international conference on Software engineering. (1996) 220–229
5. Batory, D., Johnson, C., MacDonald, B., von Heeder, D.: Achieving extensibility through product-lines and domain-specific languages: a case study. In: ACM Trans. Softw. Eng. Methodol. Volume 11. (2002) 191–214
6. Bartholet, R.G., Reynolds, P.F., Brogan, D.C., Carnahan, J.C.: In search of the philosopher's stone: Simulation composability versus component-based software design. In: Proceedings of the Fall Simulation Interoperability Workshop. (2004)
7. Carnahan, J.C., Reynolds, P.F., Brogan, D.C.: Visualizing coercible simulations. In: Proceedings of the Winter Simulation Conference, Institute of Electrical and Electronics Engineers, Inc. (2004) 411–419
8. Carnahan, J.C., Reynolds, P.F., Brogan, D.C.: Language support for identifying flexible points in coercible simulations. In: Proceedings of the Fall Simulation Interoperability Workshop. (2004)
9. Schruben, L.: Simulation modeling with event graphs. In: Commun. ACM. Volume 26. (1983) 957–963
10. Szyperski, C.: Component Software: Beyond Object-Oriented Programming. 2d edn. Addison-Wesley (2002)
11. Petty, M.D., Weisel, E.W.: A composability lexicon. In: Proceedings of the Spring Simulation Interoperabilty Workshop. (2003)
12. Page, E.H., Opper, J.M.: Observations on the complexity of composable simulation. In: Proceedings of the Winter Simulation Conference. (1999)
13. Petty, M.D., Weisel, E.W., Mielke, R.R.: Computational complexity of selecting components for composition. In: Proceedings of the Fall Simulation Interoperability Workshop. (2003)
14. Fox, M.R., Brogan, D.C., Reynolds, Jr., P.F.: Approximating component selection. In: Proceedings of the Winter Simulation Conference. (2004)
15. Page, E.H., Briggs, R., Tufarolo, J.A.: Toward a family of maturity models for the simulation interconnection problem. In: Proceedings of the Spring Simulation Interoperability Workshop. (2004)
16. Petty, M.D., Weisel, E.W.: A formal basis for a theory of semantic composability. In: Proceedings of the Spring Simulation Interoperability Workshop. (2003)
17. Weisel, E.W., Petty, M.D., Mielke, R.R.: Validity of models and classes of models in semantic composability. In: Proceedings of the Fall Simulation Interoperability Workshop. (2003)
18. Kohonen, T.: Self-Organizing Maps. Springer (1997)

The Development of Dependable and Survivable Grids*

Andrew Grimshaw[1], Marty Humphrey[1], John C. Knight[1], Anh Nguyen-Tuong[1], Jonathan Rowanhill[1], Glenn Wasson[1], and Jim Basney[2]

[1] Department of Computer Science, University of Virginia, Charlottesville, VA 22904, USA
[2] National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign, Champaign, IL, 61820, USA

Abstract. Grids should not just be facilitating advances in science and engineering; rather they should also be making an impact on our daily lives by enabling sophisticated applications such as new consumer services and support for homeland defense. This is not possible today because the poor grid dependability—which is tolerated by scientific users—would be unacceptable in critical infrastructure applications. This project aims at correcting this problem by developing technology that will allow grids to be used to provide services upon which society can depend. Through the Grid Dependability and Survivability Architecture (GDSA) and the Dependability Exchange and Specification Language (DESL), Grids will be engineered both to achieve high dependability and to permit assurance that high dependability has been achieved.

1 Introduction

Current grid systems provide services that are important to society, but our dependence will grow considerably over time reaching a point where they will become interwoven within the fabric of society. They will then have become critical infrastructure. Grid systems are dependent upon one another and will become more so. Grids providing financial services, for example, will depend on grids that support telecommunications, and grids providing health care services will depend on both financial services and telecommunications. The common need throughout all the various interdependent grid user communities is high levels of dependability; users must be able to depend upon the service they receive. This dependability will include reliability in some cases, availability in others, data integrity and confidentiality (security) in essentially all services, and possibly safety in some circumstances.

To engineer a grid to meet the expected high dependability requirements is likely to be technically infeasible or prohibitively expensive or both. Infeasibility derives from technical issues such as the reliance on commodity operating systems that are known to be vulnerable to cyber attacks. The expense derives from the very high cost of the replication of the basic components of the system, such as those providing

* This project is supported in part by the US National Science Foundation under grant SCI-0426972 (ITR for National Priorities, Network Centric Middleware Services).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 729 – 737, 2005.
© Springer-Verlag Berlin Heidelberg 2005

communications, storage and computing, that will be required if continuous availability is demanded.

The alternative approach endorsed here recognizes explicitly this fundamental trade-off between dependability and the resources and technology needed to provide it by making the next generation of grids *survivable*. Informally, a survivable system is one that provides one or more alternate services (different, less dependable, or degraded) in a given operating environment if the primary service cannot be provided because of attacks or failures. The key approach behind a survivable system is that each of these alternate services is designed to cope with potentially a different class of faults. By constructing or incorporating elements of the system (such as those implementing the primary functionality) with less provision for coping with faults than normally might be preferred, a survivable approach *may take the implementation of such a system from a complexity and cost level that is infeasible to one that is feasible*. The potential loss of service that ensues is dealt with by providing alternate services when the primary implementation is unavailable. Users receive a higher value from the system either in the form of reduced cost or increased functionality options.

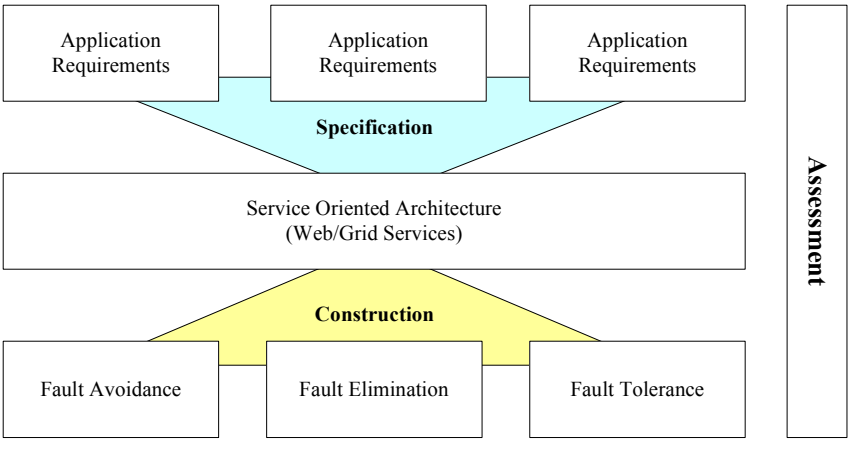


Fig. 1. Specification, construction, and assessment of dependable applications in service-oriented architectures

A survivable approach has the added advantage that it maps naturally to a development environment in which applications are built in part by incorporating existing grid elements from *different administrative domains*, an inherent characteristic of the grid paradigm. The specification of the alternate services advocated by a survivable approach provides an intellectual framework by which to analyze and incorporate existing grid elements with the inevitably different characteristics in multiple dimensions, e.g., different policies on the availability of the provided grid facilities, different expected average and peak load demands, different security policies, and different service availability guarantees. In such an environment, the dependability require-

ments for a specific application will be met in part by a process of resource discovery combined with the composition of services with the requisite levels of dependability. Note that oftentimes the straightforward composition of existing services will not be sufficient to meet the stated dependability requirements.

Our approach to the development of survivable grids is illustrated in Figure 1. At the bottom of the figure are the techniques commonly used to construct dependable systems: fault avoidance, fault elimination, and fault-tolerance. These techniques need to be enhanced to deal with the world-view that pervades anticipated grid systems and to implement the survivability concept. In the middle of the figure is the Service-Oriented Architecture (SOA). One of the hallmarks of an SOA is that of service composition: the construction of new services by composing, often dynamically, existing web services into new services. Available grid facilities change over time as circumstances change within administrative domains. Thus, for example, the hardware resources that can be provided by a specific administrative domain will depend on demands within the domain, failures that have occurred, and so on. We adapt the implementation of service provided to a given user at the time that the requirements are stated. At the top of Figure 1 are the applications, which have dependability requirements. Those requirements need to both be specified and tested against the dependability characteristics of the services the applications use.

This paper gives an overview of the key aspects of our approach. In Section 2, we describe the Dependability Exchange and Specification Language (DESL), an XML-based language used by grid elements to specify their dependability characteristics. In Section 3, we describe our dependability architecture and implementation, the Grid Dependability and Survivability Architecture (GDSA), based on a classic *data-driven* control system model. In GDSA a series of instruments will continuously monitor application and system behavior. Monitoring may be either direct polling, or subscribing to events of interest. The raw monitoring data will be processed, collected and distributed to appropriate dependability services, analyzed, and as needed both appropriate corrective actions may be taken and availability characteristics of components will be updated. GDSA and DESL build on our earlier work with Legion [1], Willow [2], OGSI.NET [3], and WSRF.NET [4][5] as well as the standards being developed in the Global Grid Forum. In particular, we will be constructing the prototype as a set of Web Services that are named with WS-References, and that fit within the context of the OGSA architecture. Section 4 contains an overall discussion on the engineering of dependable and survivable Grids. Section 5 concludes.

2 Grid Dependability and Survivability Architecture (GDSA)

GDSA is a data-driven control system architecture for dependability/survivability developed in the context of existing and evolving grid and Web Services standards, specifically the OGSA standards from the Global Grid Forum (OGSA-Program-Execution, OGSA-Core, OGSA-Security, OGSA-logging, etc.), and from the W3C and OASIS (WS-Security, WS-Context, WS-Notification, WS-Transaction, etc.).

To achieve dependability and survivability in a SOA given a dependability specification requires an architecture for detecting faults and responding to them. The problem can be thought of as a classic control system with sensors that monitor and provide status information to an analysis module that makes decisions on how to respond using an underlying control mechanism.

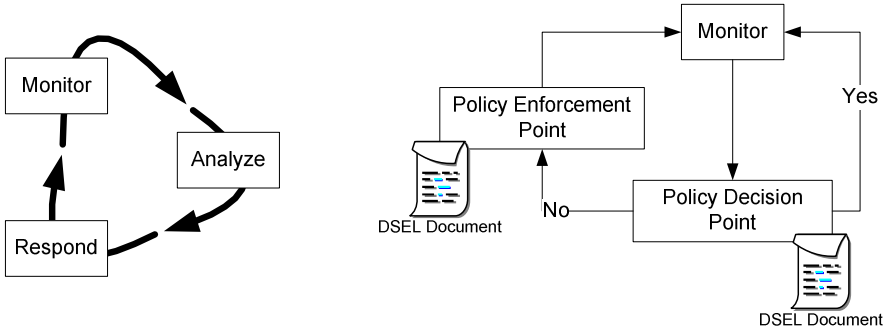


Fig. 2. Conceptual model for GDSA. The figure on the left shows the conceptual model as a classic data-driven control loop. The figure on the right shows more detail with the role of DESL as a specification language

The conceptual model is shown in Figure 2. Using various forms of monitors, the system detects errors, faults, state changes, or anomalous conditions. For example, message transmission fails because the receiver is not there, or a host fails, or an unauthorized user attempts access to critical data, or a monitor detects that the load on a host is above or below some threshold. Then analyze the events in the context of DESL specifications to see if the state with respect to the specification has changed. If conditions warrant, take some form of action to recover from the situation or exploit the change in state.

The PDP is a "policy decision point" [6]. We utilize this common terminology shared by many projects and approaches, for example XACML [7]. The PDP evaluates the current state based on information collected by the monitors or held in databases against policies and requirements carried in DESL documents. The PDP makes a "yes/no" decision. If the decision is "yes", no action is taken and the process continues. If the decision is "no" then the current state does not meet the specified requirements and action must be taken, i.e., a "policy enforcement point" (PEP) has been reached.

The PEP has as inputs the current state and a set of specifications, rules, trade-offs, and actions to take specified in a DESL document. The objective is to take an action to bring the state back to *some* correct state. We stress "some" because new correct state may not be what it was before, a degraded level of service may have been selected, or perhaps even no level of service at all.

The PDP and PEP combination can be thought of as a sort of application or service "manager" that is responsible for keeping an application or service delivering its

specified QOS. Indeed below we will often call the combination an *application manager*. Keep in mind that the "application" may in fact be the enforcement of system level property, e.g., that no priority 2 jobs run anywhere if priority 1 jobs are waiting.

Second, there will likely be many application managers running concurrently in any real system. The application managers will have different, and often conflicting, objectives. Thus, like any large set of intertwined control loops some interference is to be expected, and unexpected behaviors may emerge. This is, we believe, a fundamental property of large scale, multi-organizational, grids. Different organizations will have different objectives. We are developing this technology from the existing prototype system that is in the Willow architecture.

The monitoring aspects of GDSA are provided by the classic publish/subscribe mechanism of WS-Notification and the OGSA-Grid-Monitoring-Architecture; polling techniques that acquire system meta-data and check component "liveness"; and by the use of ExoEvents [8]. ExoEvents are a mechanism for subscribing to a set of events that may occur in the call chain of a service invocation. For example, notify monitor_A if there is a "no such service" fault. The advantage to ExoEvents over classic publish/subscribe in a grid is that the set of services that may be used in executing a particular service may not be known *a priori*, making setting up the required subscriptions difficult. Further, the subscription is often needed only during the context of a particular call - not before and not after.

The "levers" (control mechanism) in GDSA consist of the additional porttypes defined services to support dependability and survivability techniques as well as the standard services provided by OGSA. Examples of additional porttypes are save-state, transfer-state, migrate, replicate, begin-epoch, end-epoch, etc. An example of a standard OGSA service call is a call on an OGSA-container to check the status of a service or instantiate a new service instance.

3 Dependability Exchange and Specification Language (DESL)

The essence of our approach is the specification and construction of a dependability framework in the context of grid SOA's. The first critical aspect is determining how dependability requirements and characteristics will be represented, inspected, and transformed, i.e. defining a language and a set of transforms on documents in that language. We define an XML-based language, DESL, that will be used for the specification and exchange of dependability and survivability characteristics, requirements and actions. We refer to services that process DESL documents as "DESL engines".

Although the exact structure and use of DESL is still emerging, we imagine two use cases for DESL documents. First, DESL may be used as a means to export a service or application's dependability and survivability characteristics. Clients may examine these documents to determine if they wish to engage a service. Service composition engines may use these documents to determine which services to connect in order to meet the overall dependability properties of an application. DESL may also be used to express the dependability and survivability requirements that a service requires of its hosting environment or its clients. For example, a DESL document may

describe how its hosting environment should manage a group of replicants or when to rejuvenate a service [9]. In addition, DESL can describe required properties of a service's client, e.g. clients must be in the same survivability mode as the service to invoke methods on the service. In this use case, DESL can be seen as a more detailed version of WS-Policy [10].

We further refine the DESL language as used to export properties of a service. Consider the DESL document used to describe the grid service MyProxy (or a MyProxy-like service that might be available in the near term) [11].

```
<DESL xmlns:desl="http://gcg.cs.virginia.edu/DESL/02-21-04">
    <desl:Supports>
        <desl:SurvivabilityModes>
            <desl:Mode name="Normal">
                <desl:Operations>... normal delegation of credentials ... </>
            </desl:Mode>
            <desl:Mode name="NationalEmergency">
                <desl:Operations>
                    ... limited delegation creds to non-essential personnel
                </desl:Operations>
                <desl:Triggers> ... order from Dept. Homeland Security </>
                <desl:Effects>
                    ... desc. of "cost" of mode, e.g. reduced service for
some
                </desl:Effects>
            </desl:Mode>
        </desl:SurvivabilityModes>
        <desl:DependabilityInformation>
            <SoftwareProcess> ... </SoftwareProcess>
            <Availability> ... </Availability>
        </desl:DependabilityInformation>
        </desl:Supports>
    <desl:Requires>
            ... requirements of service: sub-services, performance, etc.
```

Fig. 3. A sample DESL document for MyProxy illustrating the basic document structure and the subcomponents

The DESL document in Figure 3 describes the supported "survivability modes" of the services as well as dependability properties of the service, both static (e.g. the software process used to build this service) and dynamic (e.g. the availability of the service, perhaps measured by the underlying container) in the <Supports> section. The <SurvivabilityModes> section of the document contains tags to describe the operations/actions that take place when the service is in that mode, the trigger event(s) that cause a service to get into that mode, and some description of the effects that users of the service can expect when the service is in that mode, i.e. what is the "cost" of being in a given mode. In the above example, when a national emergency is declared by the Department of Homeland Security, the MyProxy service will issue limited credentials to non-essential personnel (the effect being to prioritize grid operations from essential clients). The <Requires> section, shown for completeness, would contain any information needed by the MyProxy service from the fabric on which it depends (e.g. other services, or its underlying container) or from clients that access it.

Documents in DESL will be exposed, collected, transformed, and analyzed by "dependability services" such as the application managers above that are responsible both for making dependability assertions (based on the compositions of services that are used), for reading and interpreting assertions made by other services, and for making choices about trade-offs among multiple service implementation options.

4 Engineering Dependable and Survivable Grids

A critical obstacle to constructing dependable and survivable grids is that grid service developers are experts in neither dependable systems nor the protocols and messaging standards that run today's service-oriented grids. A supporting cast of tools, libraries and other grid services with which developers can turn their service logic into dependable and survivable grid services must be provided. To the extent possible, developers should only need to implement their application logic. However, the introduction of survivability concepts will require the mapping of survivability modes to grid services. In some cases, this mapping can be automatically performed by the container on behalf of the service, e.g., to enforce policies that limit external resource consumption or policies that can be enforced at the message level. In other cases, the developer will need to program the service to understand the various modes supported. In yet other cases, the mapping can be implemented by placing a proxy in front of existing services; this proxy would intercept a request to transition to mode X and map it to messages on the back end services.

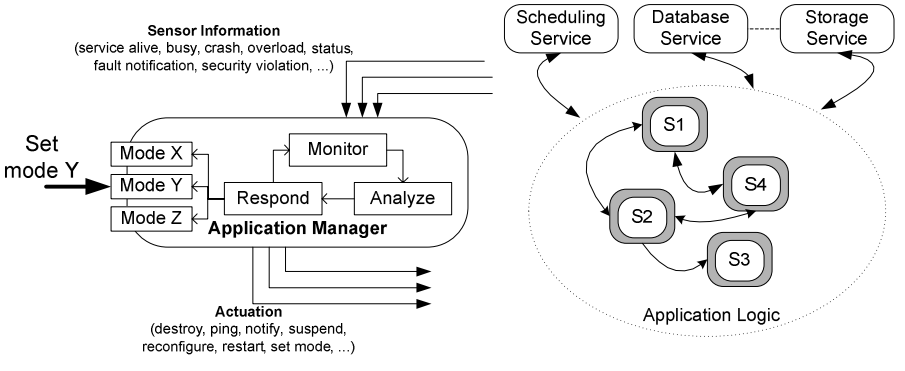


Fig. 4. Structure of a survivable grid application. Services S1-S4 represent the actual application logic. The other services provide support essential to the execution of the underlying application

Figure 4 illustrates a possible structure for a survivable grid application. The application logic is implemented by a set of communicating grid services labelled S1 through S4. The shaded region surrounding the services S1-S4 denotes that they are contained within a hosting environment, such as .net or J2EE. All other services represent the supporting cast needed to create and execute the application.

The application manager plays a vital role in the management of the application. It implements a DSEL engine and is responsible for: (a) receiving information from sensors, e.g., status information such as network and host load, network bandwidth prediction, liveness notification for S1-S4, fault notifications, security violations; (b) analyzing the stream of sensor information and taking appropriate actions, e.g., transitioning from one survivability mode to another; and then (c) issuing actuation commands, i.e., sending messages to either application services S1-S4 or other services. Note that transitions between survivability modes can be the result of external (and authorized) requests, themselves possibly issued as a result of another control loop, or as the result of an administrative decision to transition all applications within a grid domain to a given operating mode, e.g., "National Emergency".

5 Conclusion

In this paper, we have argued that a new approach is needed to construct Grids that serve as critical infrastructure. We have presented Dependability Exchange and Specification Language (DESL), used to express dependability properties and constraints, and on the overall architecture, the Grid Dependability and Survivability Architecture (GDSA). Through DESL and GDSA, Grids will be engineered both to achieve high dependability and to permit assurance that high dependability has been achieved.

References

[1] A.S. Grimshaw, A.J. Ferrari, F.C. Knabe and M.A. Humphrey, "Wide-Area Computing: Resource Sharing on a Large Scale," IEEE Computer, 32(5): 29-37, May 1999.

[2] J. C. Knight et. al., "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications," Intrusion Tolerance Workshop, DSN-2002 The International Conference on Dependable Systems and Networks, June 2002.

[3] G. Wasson, N. Beekwilder, M. Morgan, and M. Humphrey. OGSI.NET: OGSI-compliance on the .NET Framework. In Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid. April 19-22, 2004. Chicago, Illinois.

[4] Humphrey, M., G. Wasson, M. Morgan, and N. Beekwilder (2004). An Early Evaluation of WSRF and WS-Notification via WSRF.NET. *2004 Grid Computing Workshop (associated with Supercomputing 2004).* Nov 8 2004, Pittsburgh, PA.

[5] G. Wasson and M. Humphrey. Exploiting WSRF and WSRF.NET for Remote Job Execution in Grid Environments. 2005 International Parallel and Distributed Processing Symposium (IPDPS 2005), Denver CO, April 4-8, 2005.

[6] Yavatkar, R., Pendarakis, D. and Guerin, R. 2000. A Framework for Policy-based Admission Control. IETF RFC 2753. http://www.faqs.org/rfcs/rfc2753.html.

[7] Organization for the Advancement of Structured Information Standards (OASIS). Extensible Access Control Markup Language (XACML) Version 1.0. OASIS Standard, 18 February 2003. http://www.oasis-open.org/committees/xacml/

[8] Nguyen-Tuong, "Integrating Fault-Tolerance Techniques into Grid Applications," Department of Computer Science, Doctoral Dissertation, University of Virginia, August 2000.

[9] K. S. Trivedi, K. Vaidyanathan and K. Goseva-Popstojanova, "Modeling and Analysis of Software Aging and Rejuvenation, " IEEE Annual Simulation Symposium, April 2000.

[10] D. Box, et. al. Web Services Policy Framework (WS-Policy). Version of 28 May 2003. http://www-106.ibm.com/developerworks/library/ws-polfram/

[11] J. Novotny, S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

On the Fundamental Tautology of Validating Data-Driven Models and Simulations

John Michopoulos and Sam Lambrakos

Materials Science and Technology Division,
U.S. Naval Research Laboratory, Washington, DC. 20375, U.S.A
{john.michopoulos, lambrakos}@nrl.navy.mil

Abstract. Recent advances in Dynamic Data Driven Application Systems (DDDAS) facilitated by the present level of computational technologies, as well as advances in data-driven modeling and simulation, impose the need for a critical evaluation of paradigms underlying Qualification, Validation and Verification (QV&V). This paper discusses the fundamental irrelevance of conventional validation procedures with respect to data-driven models and simulations. This inherent property of data-driven models and simulations makes the data-driven approaches extremely desirable from a reliability perspective. An informal comparison of the logical flow of traditional and evolved QV&V demonstrates the tautological nature of data-driven model validation. A brief epistemological review of the origins of traditional and evolved QV&V is also presented.

1 Introduction

The need for high fidelity simulation of realistic size and complexity systems have motivated studies concerning methodologies for the development of Dynamic Data Driven Application Systems (DDDAS) by various groups. The National Science Foundation has taken a leadership role in fostering research on such efforts through the initiative on DDDAS [1].

In the past, aspects of DDDAS and dynamic data in particular, have been used to reduce the effect of uncertainties on weather prediction modeling methods, and correct error propagation to improve the fidelity of model predictions. In addition, data-driven aspects of DDDAS have also been used in the past for very successful material characterization.

Recent advances in DDDAS facilitated by the present level of computational achievements, as well as advances in data-driven modeling and simulation, impose the need for a critical evaluation of paradigms underlying Qualification, Validation and Verification (QV&V). This paper focuses on the fundamental irrelevance of conventional validation procedures with respect to data-driven models and simulations. This follows since conventional validation procedures are based on paradigms and associated terminology, that are historically biased and related to concepts and implementations that do not correspond to today's advanced computational practices and possibilities.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 738–745, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Additional areas of contemporary interest where there is a need for dynamic data-driven Modeling and Simulation (M&S) of systems are multi-physics systemic behavior prediction, multi-scale fire dynamics prediction, disaster management decision support, bio-chemical contaminant dispersion prediction, hydrogen storage, welding and heat deposition processes, artificial muscle autonomous systems, fuel cells, soft robotics and nano-robotics applications, to name only a few.

There are formal aspects of "data-driven system modeling" that are related to methods of "system identification" and "parameter estimation," which are based on formal systems theory. In the discussions that follow, the concept of system identification has been extended to include that of data-driven modeling. Further, "dynamic" data driven is to be understood as implying system identification within the context real-time input of data, which is considered as having been obtained most recently from sensors. Similarly, the use of most recently obtained data for adaptive simulation control or steering, constitutes the second use of the term "dynamic data driven" as has been discussed elsewhere [2, 3, 4].

A key aspect of our development concerns the concept of "weak termination" that implies informally the incomplete character of validation with respect to the predictive capability of a model. A rigorous examination of the mathematical foundations of this concept is beyond the scope of this presentation in that it relates to temporal and ontological formalisms of logical meta-systems [5].

In this paper, we first present the definitions and origins of the particular QV&V terms. We subsequently identify how weak termination undermines the fidelity, accuracy and reusability of models. We further demonstrate how the application of data-driven M&S avoids weak termination and enables reusability of models and simulations. We conclude with a presentation of the association of scientific methods with forward and inverse (data-driven) modeling, indicating that forward modeling is an artifact of practices based on applying the hypothetico-deductive method. These practices are not necessary when behavioral data can be collected and utilized for M&S.

2 Qualification, Validation and Verification

Model fidelity, accuracy, and high confidence in predictability requires that contemporary M&S are subject to various QV&V procedures. There are many descriptions of how QV&V relates to modeling and simulation. Figure 1 represents the modeling and simulation process along with the associated QV&V in terms of logical flow. This logical flow represents a unification of the abstractions defined by many organizations such as the AIAA [6], ASME [7], DoD's Defense Modeling and Simulation Office (DMSO) [8] and DOE Defense Program's (DOE/DP) Accelerated Strategic Computing Initiative (ASCI) [9]. The dotted arrows shown in this figure represent human activities that are implemented with various degrees of automation, allowing a transitioning from the physical system to the system's conceptual model (via analysis), next to the computational model (via computational methods), and then back to the physical system (via simulation).

A conceptual model is constructed by analysis of the behavioral structure of the physical system within an application context. The conceptual model can be a set of partial differential equations (PDEs) representing conservation laws with appropriate constitutive equations. Typically, this type of mathematical representation is known as an analytical model that encapsulates the conceptual model. This model reproduces the behavior of the physical system and belongs to the class of models designated as "explicit" or "physics-aware" [10].

Another form of conceptual model is that of a "physics-agnostic" model such as rule-based (e.g. Cellular Automata or Genetic Algorithms) or input-output associator technologies (e.g. Neural Nets, Perceptrons, Support Vector Machines). The corresponding analytical or mathematical model introduces errors associated with its underlying idealized assumptions.

The computational model is encapsulated by the software that implements the conceptual model within the computational infrastructure. It is constructed by a variety of programming techniques based on various degrees of computational automation ranging from manual to automated software generation that exploits the ability of commercial design tools. The computational model justifies the need for verification because it introduces additional uncertainty that is associated with space and time discretization, i.e. errors associated with the discrete representation of differential operators and machine truncation.

The formal definitions of the QV&V terms are as follows [6]:

- *Qualification* is the process of determining that a conceptual model implementation represents correctly a real physical system.
- *Verification* is the process of determining that a computational model implementation represents correctly a conceptual model of the physical system.
- *Validation* is the process of determining the degree to which a computer model is an accurate representation of the physical system from the perspective of the intended uses of the model.

Both validation and qualification attempt to establish the representational fidelity of the conceptual (qualification) and computational (validation) models relative to the physical system. For this reason, qualification may be considered another form of validation. This consideration explains why most of the bibliography concerns verification and validation (V&V). Verification attempts to determine the error and accuracy between two models (conceptual and computational). The modeling of uncertainty has to address the error originating from various sources extending from sensor device noise to algorithmic approximation and computational implementation of mathematical primitives by particular hardware architectures.

Shown in Fig. 1 are QV&V concepts in terms of a comparison between the behavior of the physical system (via experimentation), and the conceptual and computational systems (via simulation). In addition to complexity, the most critical issue associated with this data-driven behavioral comparison among the various system representations is "weak termination". Unfortunately, termination of the recursive process of minimizing the differences between model predicted data sets and data sets generated by the physical system, is not guaranteed for

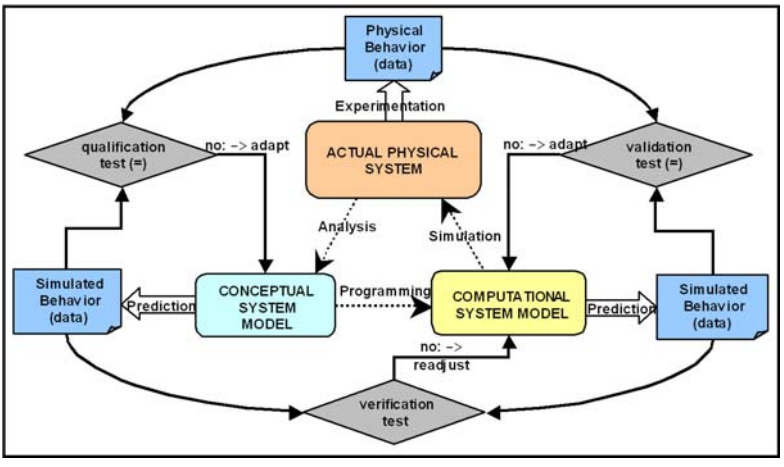


Fig. 1. Traditional QV&V for the M&S process

the user defined accuracy (as a measure of fidelity of simulation) in a finite time, nor is it guaranteed that there is a unique model prediction that converges to the physical system behavior with a desirable speed. This is represented clearly by the fact that as long as the comparison modules in Fig. 1 evaluate to "no," the methodology will continue to iterate. This situation forces the user in general, to employ "engineering approximations." That is to say, attitudes of accepting models within "acceptable bounds" for specific applications, which consequently lead to a plethora of low-confidence models that vary according to the personal modeling assumptions of the specific user.

3 Embedding Validation into Modeling and Simulation

Addressing weak termination as well as moving towards a realistic approach that guarantees relative fidelity in a model requires considering an alternative approach. This approach involves implementing an M&S methodology that can utilize: (1) *data-streams* of controlled continuous system behavior (stimulus-response pairs of all observables regardless of their field or non-field nature); (2) *analytical representations of the model* that can accurately reproduce a subset of the acquired data via system identification through successive dynamic model adaptation with the help of optimization techniques (encoding intrinsically the validity of the derived model); and (3), *derived models* for simulating the predictive response of the originally modeled system or any other system that shares the identified continuous behavior with the original system.

Figure 2 shows the proposed data driven M&S process as a modification of that shown in Fig. 1. The modules located within the dashed-line region constitute an alternative conceptual model that is defined by the optimization structure comprising these modules. This approach effectively embeds data asso-

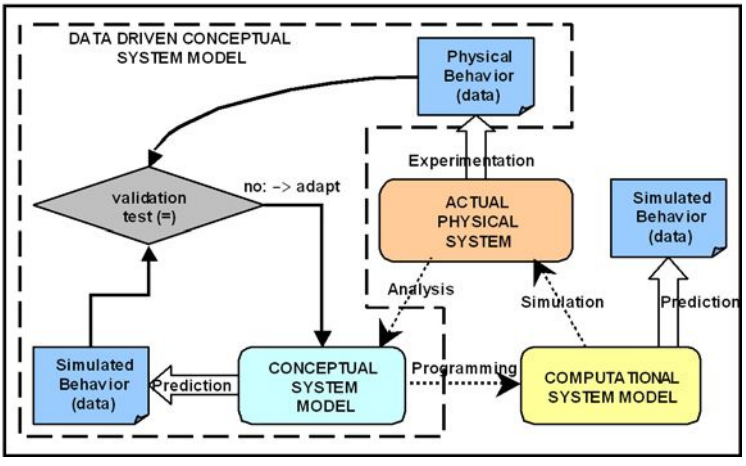


Fig. 2. Evolved QV&V for data-driven M&S process

ciated with observed behavior of the physical system, into the conceptual model, thus guaranteeing the intrinsic validity of the conceptual system. For continuous systems, this model usually refers to the mathematical formulation representing the conservation laws and associated constitutive equations. Since no comparative module exists between the physical and simulated data (see Fig. 2) this methodology eliminates entirely the weak termination problem. Termination is strong because the adaptively computed model is trained on the behavioral data and therefore must terminate if it satisfies the optimization criteria, including any acceptable error tolerance between actual and simulated data. Accordingly, termination occurs prior to using the model for prediction of behavior.

4 Epistemological Origins of QV&V in M&S

A more detailed examination of the logic underlying V&V in traditional M&S as shown in Fig. 1, indicates a paradigm for prediction that is based on the hypothetico-deductive scientific method (HDSM) [11], and that is applied by the majority of researchers. This paradigm was first applied by Newton [12, 13], who established the foundation for the Kantian reconciliation of the Cartesian rationalism with the Baconian empiricism [11]. Its intermediate stages were marked by the influence of positivism that was based on Compte's strict empiricist claims that valid knowledge is only that which is based on experience [12, 13]. The "Vienna Circle" of scientists and logicians, further contributed by establishing the logical positivism aspect of the method, which accepted Wittgenstein's [12] verification theory of meaning that considers as meaningful only those prepositions (theories) that can be empirically verified.

The HDFM program involves the steps of making an original observation of a physical system's behavior, making a hypothesis (i.e. making a highly idealized

abstract model) that may explain (predict) the observation, formulating a theory to be used for predictions, and finally generating observations via experiments for the purpose of verifying [14] or falsifying [15] the theory. If the experiment verifies the theory, then the theory can be considered valid until another experiment renders it false. Further softening of these attitudes resulted from the Logical Empiricism of Carnap [16] that replaced the idea of verification with the idea of "gradually increasing confirmation." That is to say, if the experiment falsifies the theory, then one must change the hypothesis and/or the theory and then reattempt validation. The most distinctive and highly argued [15] characteristics of a theory generated by this approach, are that it is tentative, non-permanent and iterative. These characteristics imply the presence weak termination as identified earlier for the M&S. There is a strong tendency in the literature to link Popper's falsification idea to the inductive scientific method leading to inverse modeling. From our perspective however, since weak termination is inherently present (i.e. if a contradictory experimental result appears, then the model is false), we have considered this approach as just a more robust HDSM than that of Hempel [10].

As an alternative to a strict adherence to a single scientific method, Critical Relativism advocates [17] that there is no single scientific method. Instead, disciplinary knowledge (as exercised through particular scientific methods) is to be viewed as contingent upon the beliefs, values, standards, methods and cognitive aims of its practitioners. Consequently, critical relativism rejects the aspect of positivism that implies that there is a single knowable reality to be discovered via the scientific method [18]. This relativist view permits the transcendence of the inductivist idea of Roger and Francis Bacon, that was recommended (but not practiced) by Newton and systematized by Herschel [12, 13]. Accordingly, this idea tunneled trough HDFM practices, and into the middle sixties at the Naval Research Laboratory (NRL) as an industrialized inductive pragmatic scientific method (IIPSM) [10, 19, 20].

At NRL, the advent of coupling hydraulically controlled mechanical power technology with that of computational management and control in the area of testing machines as efficient empirical fact collectors provided the industrialized character of efficiency through automation. The inductive character of the method was provided by insisting that only theories and/or models of continuum system behavior that are consistent with the facts be considered as their functional abstractions. The pragmatism of Pierce and James [12] provided the scope for such a method by enforcing the idea that what matters is not the structural elegance of a theory or model, but instead, the confidence and fidelity of prediction of a theory/model that makes it suitable for application.

The IIPSM program involves the massive automated acquisition of experimental measurements of a physical system's behavioral response. This is followed by construction of a theory in the form of a specific mathematical representation with adjustable parameters that are capable of encoding system behavior. The values of these parameters are assigned by minimizing the error between predicted and measured data sets. Finally, the theory is used for predicting system response given the condition that the stimulus to behavior (or input to the

system) is within the bounds of the original experimentally observed data-set. it follows that for this approach theory/model validation is inherent (since it has been constructed to reproduce the observed data) and therefore weak termination of its verifiability is not relevant.

In the case of HDSM we start with a small amount of data (or no data at all) that represents casual empirical observations, then we generate a plausible theory/model (a set of equations associated with a given axiomatic foundation), then we generate a predicted-behavior data set (simulation), and finally we perform experimentation to collect the actual behavior data set, in order to compare the two. This implies a logical flow from theory to prediction and then to data. In the sense that we are marching forward from theory to data, this approach has also been referred to by various investigators as the "forward," "theory-driven" or "traditional" approach. On the other hand, the IIPSM implies a logical flow from data to theory, and then to prediction. Relative to the directionality of logical flow between data and theory, this approach is inverse to the HDSM and is termed the "data-driven" approach.

5 Conclusions

Our discussion has brought to light significant aspects of validation as they relate to M&S. In particular, that validation is actually not needed by those who pursue data-driven adaptive M&S methods because it is an inherent property of these methods. This inherent validation makes data-driven methods very desirable from a reliability perspective and suggests that when enough data can be collected then data-driven methods should be considered favorably.

Acknowledgments

The authors acknowledge the support by the National Science Foundation under grant ITR-0205663 and express their gratitude to Dr. F. Darema for her support of our DDDAS-related activities. Partial support from NRL's 6.1 core-funding is also greatly acknowledged.

References

1. Darema, F., Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements. International Conference on Computational Science - ICCS'04, Krakow, Poland, June 6-9, 2004, Part III Series: Lecture Notes in Computer Science, Bubak, M.; Albada, G.D.v.; Sloot, P.M.A.; Dongarra, J. (Eds.), LNCS 3038, (2004), 662-669.
2. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Rice, J., Farhat, C., Lesoinne, M., Lechenault, F., DDEMA: A Data Driven Environment for Multiphysics Applications,in: Proceedings of International Conference of Computational Science - ICCS'03, Sloot, P.M.A., et al. (Eds.) Melbourne Australia, June 2-4, LNCS 2660, Part IV, Springer-Verlag, Haidelberg, (2003), 309-318.

3. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Farhat, C., Lesoinne, M., Rice, J., Joshi, A., On a Data Driven Environment for Multiphysics Applications, Future Generation Computer Systems, in-print (2005).
4. Michopoulos, J., Tsompanopoulou, P., Houstis, E., Joshi, A., Agent-based Simulation of Data-Driven Fire Propagation Dynamics, Proceedings, International Conference Computational Science - ICCS'04, Krakow, Poland, June 6-9, 2004, Part III Series: Lecture Notes in Computer Science, Bubak, M.; Albada, G.D.v.; Sloot, P.M.A.; Dongarra, J. (Eds.), LNCS 3038, (2004), 732-739.
5. Van Benthem, J.F.A.K, The Logic of Time, D. Reidel Publishing Company, Doordrecht, (1949).
6. American Institute of Aeronautics and Astronautics Standards Program, Guide for the Verification and Validation of Computational Fluid Dynamics Simulations, AIAA report G-077-1998, (1998).
7. ASME-JFE, Journal of Fluids Engineering Editorial Policy Statement on Control or Numerical Accuracy, Jnl. Of Fluids Engineering, Vol. 115, 3, (1993), 339-340.
8. DoD, Verification, Validation, and Accredization (VV&A) Recommended Practices Guide, Defense Modeling Simulation Office, Office of the Director of Defense Research and Engr., available: www.dmso.mil/docslib
9. M. Pilch, T. Trucano, J. Moya, G. Froelich, A. Hodges, D. Peercy, Guidelines for Sandia ASCI Verification and Validation Plans - Content and Format: Version 2.0, Sandia Reports SAN2000-3101, (2001).
10. Michopoulos, J.G., Mechatronically Automated Characterization of Material Constitutive Response, in Proceedings of the 6th World Congress on Computational Mechanics (WCCM-VI), September 5-10 2004, Beijing China, Tsinghua University Press and Springer, (2004) 486-491.
11. P. Thagard, Computational Philosophy of Science, MIT Press, Cam- bridge, MA, (1988).
12. Cooper, D.E., Epistemology: The Classic Readings, Blackwell Publ., Oxford, (1999).
13. Losee, J., A Historical Introduction to the Philosophy of Science, Oxford Un. Press, Oxford (2001).
14. C. G. Hempel, Aspects of Scientific Explanation, Free Press, New York, (1965).
15. K. Popper, Unended Quest, Fontana, London, 1976.
16. Carnap, R., The Philosophy of Science, New York, NY: Basic Books, (1966).
17. Anderson, P.F., Marketing, Scientific Progress, and Scientific Method, Journal of Marketing, 47, (Fall), (1983), 18-31.
18. Olson, J.C., Towards a Science of Consumer Behavior, in Advances in Consumer Research, Vol. 9, ed. Andrew A. Mitchell, Ann Arbor, MI: Association for Consumer Research, v- x, (1981).
19. Michopoulos, J.G., Computational and Mechatronic Automation of Multiphysics Research for Structural and Material Systems, Book Chapter, in "Recent advances in Composite Materials" in honor of A.A. Paipetis, Kluwer Academic publishing (2003).
20. Lambrakos, S.G., and Milewski, J.O., Analysis of Processes Involving Heat Deposition Using Constrained Optimization, Sci. and Tech. Of Welding and Joining, 7 (3), (2002), 137.

Managing Heterogeneity in a Grid Parallel Haskell

A. Al Zain[1], P.W. Trinder[1], H-W. Loidl[2], and G.J. Michaelson[1]

[1] School of Mathematical and Computer Sciences, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, UK
{ceeatia, trinder, greg}@macs.hw.ac.uk
[2] Ludwig-Maximilians-Universität München, Institut für Informatik, D 80538 München, Germany
hwloidl@informatik.uni-muenchen.de

Abstract. `Grid-GUM` is a distributed virtual shared-memory implementation of a high-level parallel language for computational Grids. While the implementation delivers good speedups on multiple homogeneous clusters with low-latency interconnect, on heterogeneous clusters, however, poor load balance limits performance. Here we present new load management mechanisms that combine static and partial dynamic information to adapt to heterogeneous Grids. The mechanisms are evaluated by measuring four non-trivial programs with different parallel properties, and show runtime improvements between 17% and 57%, with the most dynamic program giving the greatest improvement.

1 Introduction

Hardware price/performance ratios and improved middleware and network technologies make cluster computing and computational Grids increasingly attractive. These architectures are typically heterogeneous in the sense that they combine processing elements with different CPU speeds and memory characteristics. Parallel programming on such heterogeneous architectures is more challenging than on classical homogeneous high performance architectures.

Rather than requiring the programmer to explicitly manage low level issues such as heterogeneity we advocate a high-level parallel programming language, specifically GpH, where the programmer controls only a few key parallel coordination aspects. The remaining coordination aspects including heterogeneity are dynamically managed by a sophisticated runtime environment, GUM. GUM has been engineered to deliver good performance on classical HPCs and clusters [6]. `Grid-GUM` is a port of GUM to computational Grids using the Globus Toolkit, the de-facto standard. As `Grid-GUM` implements a virtual shared-memory over a wide-area network, it is perhaps surprising that it gives good performance in some instances, e.g. on homogeneous low-latency multiclusters [1]. However for heterogeneous architectures load management emerges as the performance-limiting issue.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 746–754, 2005.
© Springer-Verlag Berlin Heidelberg 2005

This paper presents `Grid-GUM2`, incorporating new load management mechanisms for heterogeneous architectures. The new mechanisms are decentralised, obtain complete static information during start up, and then cheaply propagate partial dynamic information during execution. The existing load management techniques in `Grid-GUM` are described in Section 2. The design of the new thread, load and communication management mechanisms is given in Section 3. The effectiveness of the new mechanisms for heterogeneous clusters is investigated using four non-trivial programs from a range of application areas, and with varying degrees of irregular parallelism and using both data parallel and divide-and-conquer paradigms in Section 4. Related work is discussed in Section 5, and we conclude in Section 6.

2 GpH and Grid-GUM

GpH (*Glasgow parallel Haskell*) [7] is a modest and conservative extension of Haskell 98 with very high level coordination, i.e. control of parallel execution. Parallel and sequential composition primitives introduce and synchronise threads. Evaluation strategies are polymorphic higher-order functions that abstract over the primitives to provide parameterisable, reusable high level coordination control.

Grid-GUM `Grid-GUM` extends the existing `GUM` memory management, and thread management techniques. In particular, it implements a virtual shared heap over a wide-area network [8]. The communication management in `Grid-GUM` is similar to `GUM`, but uses MPICH-G2, a Grid-enabled implementation of the MPI standard [5], and hence the Globus Toolkit as middle-ware.

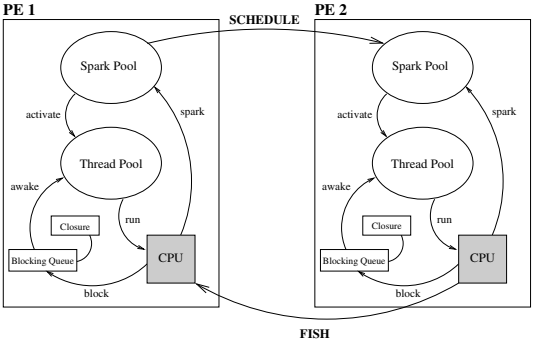


Fig. 1. Interaction of the components of a GUM processing element

`Grid-GUM` uses an *evaluate-and-die* thread management model with *sparks*, i.e. pointers to graph structures, to represent potential parallelism. Sparks are generated by an explicit `par` construct in the program and maintained by the runtime-system in a flat *spark pool*. A sparked expression may be executed by an independent thread. However, if a thread needs the value of the expression, and

no other thread is evaluating it, the demanding thread will perform the computation itself. This behaviour is called *thread subsumption* because the potentially parallel work is inlined by another thread.

Figure 1 illustrates the load management mechanism in `Grid-GUM`, depicting the logical components on each Processing Element (PE) of the `Grid-GUM` abstract machine. When activated a spark causes a new thread to be generated. Threads that are not currently being executed reside in the thread pool. When the CPU is idle, and the thread pool is empty, a spark will be activated to generate a thread. If a running thread blocks on unavailable data, it is added to the blocking queue of that node until the data becomes available.

The thick arrows between the PEs in Figure 1 show load management messages exchanged in `Grid-GUM`. Initially all processors, except for the main PE, will be idle, with no local sparks available. PE2 sends a `FISH` message to a random-chosen PE. On arrival of this message, PE1 will search for a spark and, if available, send it to PE2. This mechanism is usually called *work stealing* or *passive load distribution*, since an idle processor has to ask for work. `Grid-GUM` also improves load distribution by using *Limited Thread* mechanism which includes specifying a hard limit on the total number of live threads, i.e. runnable or blocked threads in the thread pool. To summarise, the `Grid-GUM` load mechanism deals with both locating work (Figure 2.a), and handling work requests (Figure 2.b), where these activities are performed in the main scheduler loop between thread time slices.

```
IF idle THEN
  IF runnable thread THEN
    execute runnable thread
  ELSE
    IF spark in spark pool then
      create runnable thread
      execute runnable thread
    ELSE
       send fish to random PE
```

Fig.2a. Work locating

```
IF received fish THEN
  IF sparks available THEN
    send spark
      to fishing PE
  ELSE
    forward fish to random PE
```

Fig.2b. Work request handling

Fig. 2. `Grid-GUM` Load Management

`Grid-GUM` **performance** In earlier work we have investigated the performance of GpH programs on `Grid-GUM`, with varying numbers of clusters and interconnection latencies [1]. We find that `Grid-GUM` delivers good and predictable speedups for several configurations, e.g. multiple homogeneous clusters with a low-latency interconnect, or programs that perform little communication on multiple clusters with a high latency interconnect. However other configurations, including heterogeneous clusters, `Grid-GUM` gives poor performance due to poor load management.

3 Grid-GUM2

Distributed load management assigns processes (or threads) to PEs in a multi-PEs system and can be classified as depicted in Figure 3. In *static* load management, PEs are assigned tasks at compile time, i.e. before the execution begins. In contrast, *dynamic* load management assumes that limited knowledge about the processes and PEs is available *a priori*, and load management decisions are made during execution. Dynamic scheduling is subdivided into two categories: a) *centralised* where a single PE is responsible for scheduling, and b) *decentralised* where all PEs participate in the scheduling. Distributed techniques can either operate in a *passive* mode where idle PEs request work, or in an *active* mode where PEs are assigned work without a prior request. This section presents

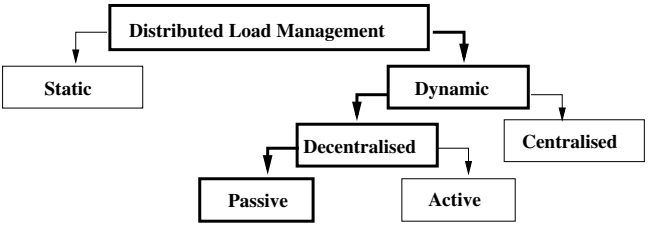


Fig. 3. Classifying `Grid-GUM2` Load Management

`Grid-GUM2`, a new adapted load scheduling to improve `Grid-GUM` performance. `Grid-GUM2` load management is dynamic, decentralised and mostly passive using complete static and partial dynamic information about PEs, processes and network. The salient features of `Grid-GUM2` are as follows.

Targeted Load Management: The challenge in load managment is to efficiently and effectively distribute the available work using dynamic information about PE CPU speed and loads, with minimal overheads. In `Grid-GUM`, if a PE is idle, it sends a *FISH* message to a randomly-chosen PE. However, given the increased heterogeneity of computational Grids, `Grid-GUM2` sources work from a PE that has a high load relative to its speed.

Load information must be maintained dynamically, but it is prohibitively expensive to regularly broadcast complete load information to every PE in a large computational Grid. `Grid-GUM2` uses a lightweight approach maintaining partial dynamic PE information at almost no cost. Each PE maintains a table *PEDynamic* that contains partial timestamped dynamic information about the PEs, currently just their loads as depicted in Figure 5. The PEDynamic table is included with every message sent between PEs, and the receiving PE updates its table with more recent information. The additional cost of sending the table is minimal for the high bandwidth interconnects in typical computational Grid.

Specialised Thread Management: `Grid-GUM` has a parameter that specifies the minimum number of sparks that should be available in each PE, the *low-*

watermark. If the number of sparks in a PE falls below the low-watermark it issues a FISH message to obtain more work. Such a uniform strategy for all PEs must be adapted for heterogeneous clusters. Our design is to maintain a local low-watermark for each PE, with higher watermarks for fast PEs. This encourages fast PEs to be more aggressive about obtaining work than slower PEs. However, local low watermarks are not get implemented in the system measured in the next section.

Variable Latency Communication Management: The interconnects between PEs in a computational Grid have varying static and dynamic latencies, e.g. the latency between PEs in the same cluster is much less than that between PEs on different clusters, and both are affected by network traffic. Our

PE	CPU Speed	Time Stamp
A	550 MHz	13:40:01
D	550 MHz	13:45:00
C	350 MHz	12:40:03
B	350 MHz	13:44:03
F	350 MHz	14:40:03

Fig. 4. PEStatic Table

PE	Load	time_stamp
A	2000	14:13:49
D	3000	14:13:59
B	10000	14:12:22
⋮	⋮	⋮

Fig. 5. PEDynamic Table

PE	Latency	Last Update
F	0.75 msec	12:45:20
G	2.05 msec	12:24:50
C	10.00 msec	12:50:25
⋮	⋮	⋮

Fig. 6. Communic. Table

mechanism prefers to obtain work and hence data from PEs that currently have low communication latency. `Grid-GUM2` maintains communication information at almost no cost. Each PE has a table *Communication* that contains partial timestamped dynamic information about the communication, currently just the latency to each PE as depicted in Figure 6. The latency of every message is estimated, timestamped and recorded in the Communications table with the startup messages initialising the table. The problems of maintaining synchronised distributed clocks is minimised by measuring message send and receive times as time elapsed since the program start.

`Grid-GUM2` introduces a novel mechanism to deal with cases where there is no work available on local, i.e low-latency, PEs. When an idle PE requests work from a PE residing outside its cluster, then a scheduling decision is automatically taken on the basis of minimising intra-cluster communication. One can distinguish two scenarios: a) if the request originated from relatively powerful cluster, then multiple sparks, i.e. several work items, are returned in a *Super-Schedule* message; b) if the request originated from a relatively weak cluster then the request is served as usual, i.e. as a single spark. Note that the idea of `Super-Schedule` is not evaluated in this work since it is under implementation.

The core of `Grid-GUM2` load management can be summarised as work location (Figure 3.a), and work request handling (Figure 3.b).

```
IF idle THEN
 IF sparks < local-watermark THEN
   send fish+local data
     to busiest PE from tables
  IF runnable-thread THEN
    execute runnable-thread
  IF spark in the spark-pool THEN
    create runnable-thread
    execute runnable-thread
  ELSE
    send fish+local data
      to busiest PE from tables
```

Fig.3.a. Work locating

```
IF received fish THEN
  update tables with data
    from fishing PE
  IF sparks available THEN
    IF fishing PE is local THEN
      send spark in schedule
        to fishing PE+local data
    Else
      send spark(s) in super-schedule
        to fishing PE+local data
  ELSE
    IF another PE has spark
      forward fish+local data
        to busiest local PE
```

Fig.3.b. Work request handling

Fig. 7. `Grid-GUM2` Load Management

4 Performance Comparison

Table 1 compares the runtime for four GpH programs with different paradigms, and parallelism regularities, on multiple heterogeneous clusters with moderate latency interconnect. The four programs are measured in this experiment are: `queens` which places chess pieces on a board; `sumEuler` which computes the sum of the Euler totient values of a list; `linSolv` which finds an exact solution of a linear system of equations; `raytracer` which calculates a 2D image of a given scene of 3D objects by tracing all rays in a given grid, or window. All run-times are the median of three executions to ameliorate the impact of the operating system and shared network interaction. The experiments were performed on eight machines: four machines with fast CPU speed (1395 MHz) and four with slow CPU speed (534 MHz).

Overall, the dynamic adaptive scheduling of `Grid-GUM2` consistently shows the best performance on heterogeneous Grid multi-clusters.

Due to limited space we focus on investigating the behaviour of one benchmark program, `raytracer`. In particular, `raytracer` has highly irregular execution, and consequently is very sensitive to changes in parallel environment.

Table 1. Programs characteristics and performance

Program	Application Area	Paradigm	Regularity	Runtimes & Improvement %		
				`Grid-GUM` Basic	`Grid-GUM` + Thread Limi.	`Grid-GUM2`
`queens`	AI	Div-Conq.	Regular	668	333 50%	310 53%
`sumEuler`	Nume. Analysis	Data Para.	Limit irreg.	570	343 39%	279 51%
`linSolv`	Symb. algebra	Data Para.	Limit irreg.	217	306 −40%	180 17%
`raytracer`	Graphic	Data Para.	High irreg.	1340	814 39%	572 57%

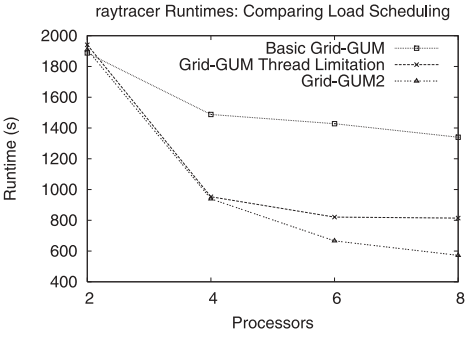
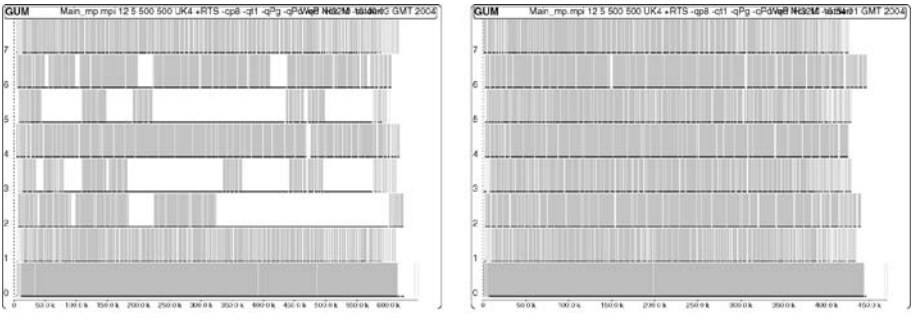


Fig. 8. Runtimes for `raytracer`



a) `Grid-GUM` b) `Grid-GUM2`

Fig. 9. per-PE activity profile for `raytracer`

Figure 8 compares the runtime achieved for the `raytracer` using different load management mechanisms on a heterogeneous environment, where there are the same numbers of slow and fast machines. For more than 4 PEs the best results are achieved for `Grid-GUM2` with better scalability and decreasing runtime than basic `Grid-GUM` and `Grid-GUM` with thread limitation.

Figure 9 shows per-PE activity profile for `raytracer`, with execution on four fast machines (0,2,4,6), and four slow machines (1,3,5,7). A per-PE activity profile shows the behaviour for each of the PEs (y-axis) over execution time (x-axis). Each PE is visualised as a horizontal line, with darker shades of gray indicating a larger number of runnable threads. Gaps in the horizontal lines indicate idleness. Figure 9.a depicts the performance on `Grid-GUM` using a limit of 1 for the number of live threads. This small limit is chosen because it is the only one that has any impact on a heterogeneous architecture. Figure 9.b depicts the performance on `Grid-GUM2`. All PEs in Figure 9.b are uniformly loaded, and finish at the same time, in contrast to the PEs in Figure 9.a have numerous idle periods. Figure 9.a also shows long idle periods at the middle of the computation, where only a small amount of parallelism is available. Here, blocking on data

that is evaluated in the PEs with slow CPU speed will cause the PEs with fast CPU speed to remain idle until new work is obtained. Note that the runtime drops from 814 s to 572 s, which almost (30%) improvement.

5 Related Work

The most closely related work is the ConCert system [4], which translates a subset of ML to machine code, for execution on a Grid architecture. In contrast to our work, parallelism is expressed via explicit synchronisation. Alt *et al* apply skeletons to computational Grids [3]. This work focuses on providing the application user with skeletons to capture common patterns of Grid abstractions. However, our aim is to provide more general programming language support for parallelism through an implementation that incorporates new implicit dynamic coordination-management strategies. Aldinucci *et al* also apply skeletons to computational Grids [2].This work focuses on providing a skeleton to centralise load management in the Grid environment. However, our aim is to solve load scheduling on the Grid by developing a dynamic decentralised load schedule.

6 Conclusion

The `Grid-GUM2` extension to the `Grid-GUM` runtime environment has been produced to efficiently and automatically manages data and work on a multi-cluster Grid environment. Achieving good performance was made possible by using a high-level parallel language and completely managing distribution at the runtime environment level . As future work we plan to measure the behaviour of `Grid-GUM2` using varying number of clusters and interconnect latencies. We are also considering building an analytic model of `Grid-GUM2`.

References

1. A. Al Zain, P. Trinder, H-W. Loidl, and G. Michaelson. Grid-GUM: Towards Grid-Enabled Haskell. In *IFL'04 — Intl. Workshop on the Implementation of Functional Languages*, Draft Proceedings, Lübeck, Germany, September 2004.
2. M. Aldinucci, M. Dnelutto, and Dünnweber. Optimization Techniques for Implementing Parallel Sckeletons in Grid Environments. In *CMPP'04 — Intl. Workshop on Constructive Methods for Parallel Programming*, Stirling, Scotland, July 2004.
3. M. Alt, H Bischof, and S. Gorlatch. Program Development for Computational Grids Using Skeletons and Performance Prediction. In *CMPP'02 — Int. Workshop on Constructive Methods for Parallel Programming*. Dagstuhl, Berlin, June 2002.
4. B-Y. Evan Chang, K. Crary, M. DeLap, R. Harper, J. Liszka, T. Murphy VII, and F. Pfenning. Trustless Grid Computing in ConCert. In *In Proceedings of the GRID 2002 Workshop*, volume 2536 of LNCS. Springer-Verlag, 2001.
5. N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A grid-enabled implementation of the message passing interface. *J. of Parallel and Distributed Computing*, 2003.

6. H-W. Loidl, F. Rubio, N. Scaife, K. Hammond, S. Horiguchi, U. Klusik, R. Loogen, G. J. Michaelson, R. Pe na, Á. J. Rebón Portillo, S. Priebe, and P. W. Trinder. Comparing Parallel Functional Languages: Programming and Performance. *Higher-order and Symbolic Computation*, 16(3), 2003.
7. P.W. Trinder, K. Hammond, H-W. Loidl, and S.L. Peyton Jones. Algorithm + Strategy = Parallelism. *J. of Functional Programming*, 8(1):23–60, January 1998.
8. P.W. Trinder, K. Hammond, J.S. Mattson Jr., A.S. Partridge, and S.L. Peyton Jones. GUM: a Portable Parallel Implementation of Haskell. In *PLDI'96 — Conf. on Programming Language Design and Implementation*, Philadephia, USA, 1996.

An Efficient Equi-semi-join Algorithm for Distributed Architectures

M. Bamha and G. Hains

LIFO, Université d'Orléans, B.P. 6759, 45067 Orléans Cedex 2, France
{bamha, ghains}@lifo.univ-orleans.fr

Abstract. Semi-joins is the most used technique to optimize the treatment of complex relational queries on distributed architectures. However the overcost related to semi-joins computation can be very high due to data skew and to the high cost of communication in distributed architectures. In this paper we present a parallel equi-semi-join algorithm for shared nothing machines. The performance of this algorithm is analyzed using the BSP cost model and is proved to have *asymptotic optimal complexity* and *perfect load balancing* even for highly skewed data. This guarantees unlimited scalability in all situations for this key algorithm.

1 Introduction

Equi-join is an expensive and frequently used operation whose parallelization is highly desirable. The *equi-join* of two tables or relations R and S on attribute A of R and attribute B of S is the relation T, written $R \bowtie S$, containing the pairs of tuples from R and S for which $R.A = S.B$. The `equi-semi-join` of S by R is the relation $S \ltimes R$ composed of the tuples of S which occur in the equi-join of R and S. Equi-semi-join reduces the size of relations to be joined and satisfies $R \bowtie S = R \bowtie (S \ltimes R) = (R \ltimes S) \bowtie (S \ltimes R)$. The equi-semi-join is generally called semi-join, and throughout the paper, semi-join refers to equi-semi-join.

The semi-join is useful in relational databases for reducing the time for processing queries involving join operations, by means of selecting only relevant data and thereby reducing the size of operand relations. Semi-join is also used to reduce the amount of data transferred over the network and therefore the communication costs in distributed architectures [10, 8]. However efficient semi-join is very sensitive to the problem of data skew. Data skew can have a disastrous effect on performance [3, 2, 1, 8, 7, 6, 11, 5] due to the high costs of communications and synchronizations in distributed architectures [2, 1, 10].

Many algorithms have been proposed to handle data skew for join and semi-join operations [8, 7, 6, 10]. Such algorithms are not efficient for many reasons:

- the presented algorithms are not scalable (and thus cannot guarantee linear speedup) because their routing decisions are generally performed by a coordinator processor while the other processors are idle,
- they cannot avoid load imbalance because they base their routing decisions on incomplete or statistical information,

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 755–763, 2005.
© Springer-Verlag Berlin Heidelberg 2005

– they cannot solve data skew problem because data redistribution is generally based on hashing data into buckets and hashing is known to be inefficient.

In this paper, we present a new parallel semi-join algorithm for shared nothing machines. This algorithm guarantees a perfect balancing of the load of the different processors during all the stages of the semi-join computation because the data redistribution is carried out jointly by all processors. Each processor deals with the data associated with a subset of the join attribute values (organized as histograms) not necessarily its "own" values. Our algorithm uses histogram's redistribution to avoid imbalance during semi-join parallel processing. Its performance is analyzed using Bulk-synchronous parallel (BSP) cost model [9] which allows us to guarantee that histogram management has a negligible cost when compared to the efficiency gains it provides to reduce the communication cost and to avoid load imbalance between processors.

2 Histograms and Semi-joins Processing: A New Approach

In this section, we present a new approach to compute semi-joins on shared nothing machines. This approach is insensitive to data skew, and is based on an efficient technique for histograms computation, allowing to reduce the *net* communication costs to a minimum.

Before carrying out the description of the semi-join algorithm, we present some results related to histograms computations which we will use thereafter. An upper bound of the global BSP cost is also given for each step.

We first assume that relation R (resp. S) is partitioned among processors by horizontal fragmentation and the fragments R_i for $i = 1, .., p$ are almost of the same size, i.e. $|R_i| \simeq \frac{|R|}{p}$ where p is the number of processors.

In the rest of this paper, we use the following notation for each relation $T \in \{R, S\}$:

– T_i denotes the fragment of relation T placed on processor i,
– $Hist(T)$ denotes the histogram of relation T with respect to the join attribute value, i.e. a list of pairs (v, n_v) where $n_v \neq 0$ is the number of tuples of relation T having the value v for the join attribute,
– $Hist(T_i)$ denotes the histogram of fragment T_i,
– $Hist_i(T)$ is processor i's fragment of the histogram of T,
– $\|T\|$ denotes the number of tuples of relation T, and
– $|T|$ denotes the size (expressed in bytes or number of pages) of relation T.

It will be assumed in the remainder of the article that, for a relation T, $\|T\|$ is sufficiently large with respect to p and in the rest of the paper, g refers to BSP communication's parameter and l the cost for a barrier of synchronization [9].

Proposition 1. *Let $Hist(R_i)_{i=1,..,p}$ be the histograms, according to some attribute A, of the fragments R_i of a relation R. If $\mathcal{H}$ is a hash function allowing*

to distribute in a balanced way the distinct values of the attribute A among the p processors, then the fragments $Hist_i(R)_{i=1,..,p}$ can be computed in time:

$$O\Big(\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + l\Big).$$

Proof: The proof proceeds according to frequency distribution of the attribute A of the relation R.

1st case: All the values of the attribute A appear in the relation R with a frequency equal to 1, i.e. the number of tuples of the relation R is equal to the number of distinct values of the attribute A (and thus $\|Hist(R)\| = \|R\|$).

In this case, the application of the hash function $\mathcal{H}$, distributes the tuples of relation R (and thus the histogram of R) evenly on the various processors. Consequently, the redistribution of $Hist(R_i)$, using the function $\mathcal{H}$, is done in time:

$$O\Big(g * \frac{|Hist(R)|}{p} + \frac{\|Hist(R)\|}{p} + l\Big) \quad \leq \quad O\Big(g * \frac{|R|}{p} + \frac{\|R\|}{p} + l\Big). \tag{1}$$

The term $g * \frac{|Hist(R)|}{p}$ is due to the fact that each processor receives a fragment of size about equal to $\frac{|Hist(R)|}{p}$, g is BSP communication parameter. The term $\frac{\|Hist(R)\|}{p}$ is time necessary to form $Hist_i(R)_{i=1,..,p}$ starting from the blocks received by each processor, during the redistribution of the local histograms $Hist(R_i)_{i=1,..,p}$ and l is the cost of a barrier of synchronization.

Inequality (1) is generally true, owing to the fact that $|Hist(R)| \leq |R|$ since the histogram contains only pairs of the form (v, n_v) where v are distinct values of attribute A of R and n_v the corresponding frequencies. So the histograms creation can be carried out in time:

$$\begin{aligned} &O\Big(\min(g * \frac{|Hist(R)|}{p} + \frac{\|Hist(R)\|}{p},\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + l\Big) \\ &\quad \leq \quad O\Big(\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + l\Big). \end{aligned} \tag{2}$$

2nd case: All the values of the attribute A appear in relation R with a frequency lower than p. In this case the relation R can be partitioned into p relations R^1, R^2, ..., R^p so that the frequency of each value v in a relation R^j is exactly equal to 1. By applying the results of the 1^{st} case to each relation R^j, we deduce that the creation of the fragments $Hist_i(R^j)$ of a relation R^j for $j = 1, .., p$ requires at most:

$$O\Big(\min(g * \frac{|Hist(R^j)|}{p} + \frac{\|Hist(R^j)\|}{p},\ g * \frac{|R^j|}{p} + \frac{\|R^j\|}{p}) + l\Big),$$

and thus the creation of the fragments $Hist_i(R)$ of relation R requires:

$$\begin{aligned} &O\Big(\sum_{j=1,..p} \min(g * \frac{|Hist(R^j)|}{p} + \frac{\|Hist(R^j)\|}{p},\ g * \frac{|R^j|}{p} + \frac{\|R^j\|}{p}) + l\Big) \\ &\quad \leq \quad O\Big(\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + l\Big), \end{aligned} \tag{3}$$

because the fragment $Hist_i(R)$ can be created starting from the fragments $Hist_i(R^j)_{j=1,..,p}$ by a simple fusion of the histograms $Hist_i(R^j)_{1\leq j\leq p}$ in time:

$$O(\sum_{j=1}^{p} \frac{\|Hist_i(R^j)\|}{p}) \leq O(\max_{i,j}(\|Hist_i(R^j)\|)) \leq O(\|Hist(R)\|),$$

and that $\sum_i \min(a_i, b_i) \leq \min(\sum_i a_i, \sum_i b_i)$. We thus combine the p phases of communication in only one (the term is not multiplied by p).

The distribution of the relation R into p relations is carried out only to show the validity of the result. In practice, if a value v appears in R_i with a frequency k, the application of the hash function to the histogram sends the pair (v, k) to the processor destination in only one message and not in k messages of the form $(v, 1)$. This reduces the amount of data exchanged between the processors.

3rd case: All the values of the attribute A appear in the relation R with frequencies higher than p. In this case, for a given value v, the tuples associated with v can be distributed on several processors and in the worst case distributed on p processors. Thus the application of the hash function, $\mathcal{H}$, to the histograms $Hist(R_i)_{i=1,..,p}$, sends pairs of the form $(v, k_i)_{i=1,..,p}$ to the same processor.

As each processor deals with a block of size $\frac{\|Hist(R)\|}{p}$, it receives at most $p * \frac{|Hist(R)|}{p} = |Hist(R)|$ data from the other processors, and the cost of the redistribution of the histograms $Hist(R_i)$ is about:

$$O\Big(g * |Hist(R)| + l\Big) \quad \leq \quad O\Big(g * \frac{|R|}{p} + l\Big). \tag{4}$$

Inequality (4) holds because all the values of attribute A appear in R with a frequency higher than p and thus: $\|Hist(R)\| \leq \frac{\|R\|}{p}$ and $|Hist(R)| \leq \frac{|R|}{p}$.

$\|Hist(R)\|$ is the time necessary to form, $Hist_i(R)$, on processor i, starting from the fragments of $Hist(R_j)_{j=1,..,p}$ received after the redistribution phase using the hash function $\mathcal{H}$. And thus the creation of the histograms $Hist_i(R)$ requires:

$$O\Big(\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + l\Big).$$

4th case: the general case follows immediately from the three preceding cases. We partition relation R into two disjoint sub-relations R' and R'' so that the tuples of the relation R' (resp. R'') are associated with frequencies lower (resp. higher) than p for the attribute A. Thus, using the first two cases, the histogram R' can be carried out in time:

$$O\Big(\min(g * |Hist(R')| + \|Hist(R')\|,\ g * \frac{|R'|}{p} + \frac{\|R'\|}{p}) + l\Big), \tag{5}$$

and the creation of the histogram of R'', (cf. 3rd case), in time:

$$O\Big(\min(g * |Hist(R'')| + \|Hist(R'')\|,\ g * \frac{|R''|}{p} + \frac{\|R''\|}{p}) + l\Big) \tag{6}$$

By making the sum of the terms 5 and 6, we show that the creation of the histograms $Hist_i(R)$ requires at most:

$$O\Big(\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + l\Big). \tag{7}$$

This is due to the fact that $Hist(R) = Hist(R') \cup Hist(R'')$ and that $\min(a, b) + \min(c, d) \leq \min(a + c, b + d)$, which proves the required result.

■

Previous parallel algorithms have used hashing functions to redistribute the relations themselves, not just their histograms. This can lead to imbalance and disastrous performance.

Remark 1. In practice, the imbalance of the data related to the use of the hash functions can be due to:

- *a bad choice of the hash function* used. This imbalance can be avoided by using the hashing techniques presented in the literature making it possible to distribute evenly the values of the join attribute with a very high probability [4]. In our algorithm, the hash functions we use may be data dependent and stored in the database index. Otherwise, for a given set of n distinct values we can find, in time of $0(n)$, a function which redistributes evenly the n values over p buckets.
- *an intrinsic data imbalance* which appears when some values of the join attribute appear more frequently than others. By definition a hash function maps tuples having the same join attribute values to the same processor. There is no way for a clever hash function to avoid load imbalance that results from these repeated values [5]. But this case cannot arise here owing to the fact that histograms contain only distinct values of the join attribute and the hashing functions we use are always applied to histograms.

In the following proposition, we present a method allowing to compute the semi-joins efficiently starting from the histograms. We show thereafter that this technique of semi-joins computation combined with the methods of histograms creation guarantees an asymptotic optimal cost for the semi-joins computation.

Proposition 2. *Let $Hist(R_i)_{i=1,..,p}$ (resp. $Hist(S_i)_{i=1,..,p}$) be the histograms, according to some attribute A, of sub-relation R_i (resp. S_i) of R (resp. S).*
If $\mathcal{H}$ is a hash function allowing to distribute in a balanced way, the distinct values of the attribute A among the p processors, then the two semi-joins, $R \ltimes S$ and $S \ltimes R$, can be computed in time:

$$\begin{aligned} Time_{sj} = O\Bigg(& \min\Big(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}\Big) + \max_{i=1,..,p} \|R_i\| \\ & + \min\Big(g * |Hist(S)| + \|Hist(S)\|,\ g * \frac{|S|}{p} + \frac{\|S\|}{p}\Big) + \max_{i=1,..,p} \|S_i\| + l\Bigg). \end{aligned}$$

Proof: To perform the semi-joins in at most the above time, the semi-joins computation, $R \ltimes S$ and $S \ltimes R$, is done in five stages.

a. A stage of creation of $Hist_i(R)_{i=1,..,p}$ (resp. $Hist_i(S)_{i=1,..,p}$) of R (resp. S): The fragments $Hist_i(R)_{i=1,..,p}$ creation is carried out by hashing the local histograms $Hist(R_i)_{i=1,..,p}$ using the hash function $\mathcal{H}$. On each processor i, the application of the function $\mathcal{H}$ distributes $Hist(R_i)$ into p buckets $Hist^j(R_i)_{i=1,..,p}$ where $Hist^j(R_i)$ is the fragment of $Hist(R_i)$ which will be sent to processor j. By using Proposition 1, we have: $|\cup_i Hist^j(R_i)| \leq \min(|Hist(R)|, \frac{|R|}{p})$.

The fragments $Hist^j(R_i)_{i=1,..,p}$ are then gathered, on each processor j, to form $Hist_j(R)$ in time: $O(\min(\|Hist(R)\|, \frac{\|R\|}{p}))$.
Hence the creation of the histograms $Hist_i(R)_{i=1,..,p}$ of relation R requires:

$$O\Big(\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \tfrac{|R|}{p} + \tfrac{\|R\|}{p})\Big).$$

Similarly for relation S, the above technique is used to compute the fragments $Hist_i(S)_{i=1,..,p}$ in time: $O\Big(\min(g * |Hist(S)| + \|Hist(S)\|,\ g * \frac{|S|}{p} + \frac{\|S\|}{p})\Big)$.

Therefore, the creation of $Hist_i(R)_{i=1,..,p}$ and $Hist_i(S)_{i=1,..,p}$ requires:

$$\begin{aligned} Time_{\text{step.a}} = O\Big(&\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) \\ &+ \min(g * |Hist(S)| + \|Hist(S)\|,\ g * \frac{|S|}{p} + \frac{\|S\|}{p}) + l\Big). \end{aligned}$$

b. A stage of creation of the fragments $Hist_i(R \bowtie S)_{i=1,..,p}$:
$Hist_i(R \bowtie S)$ is computed, on each processor i, by intersecting $Hist_i(R)$ and $Hist_i(S)$ (i.e. by calculating $\{v | v \in Hist_i(R)\} \cap \{v | v \in Hist_i(S)\}$), in time:

$$Time_{\text{step.b}} = O\Big(\max_{i=1,..,p}(\min(\|Hist_i(R)\|, \|Hist_i(S)\|))\Big).$$

c. We then compute, on each processor j, the intersections:

$$\begin{cases} \overline{Hist}^j(R_i) = Hist^j(R_i) \cap Hist_j(R \bowtie S) \text{ for } i = 1,..,p \text{ and} \\ \overline{Hist}^j(S_i) = Hist^j(S_i) \cap Hist_j(R \bowtie S) \text{ for } i = 1,..,p \end{cases}$$

in time: $O\Big(\sum_i \|Hist^j(R_i)\| + \sum_i \|Hist^j(S_i)\|\Big)$.

In accordance with stage a., one has: $\begin{cases} \sum_i \|Hist^j(R_i)\| \leq \min\Big(\|Hist(R)\|, \frac{\|R\|}{p}\Big) \\ \sum_i \|Hist^j(S_i)\| \leq \min\Big(\|Hist(S)\|, \frac{\|S\|}{p}\Big), \end{cases}$
and thus the total cost of this stage is:

$$Time_{\text{step.c}} = O\Big(min(\|Hist(R)\|, \frac{\|R\|}{p}) + min(\|Hist(S)\|, \frac{\|S\|}{p})\Big).$$

d. A stage of communication where each processor i sends each fragment $\overline{Hist}^j(R_i)$ to processor j: During this stage of communication, each processor i, receives $\sum_j |\overline{Hist}^j(R_i)|$ pages of data from the other processors in time

$O(\sum_j g * |\overline{Hist}^j(R_i)| + l)$. We point out that: $Hist(R_i) = \cup_j Hist^j(R_i)$ and that: $|Hist(R_i)| = \sum_j |Hist^j(R_i)| \geq \sum_j |Hist^j(R_i) \cap Hist(R \ltimes S)|$
and thus $|Hist(R_i)| \geq \sum_j |\overline{Hist}^j(R_i)|$. Consequently the total cost of this stage is at most: $Time_{\text{step.d}} = O\Big(g * |\overline{Hist}(R_i)| + l\Big)$.

Remark 2. *$\cup_j \overline{Hist}^j(R_i)$ is simply the intersection of $Hist(R_i)$ and $Hist(R \ltimes S)$. This intersection is noted: $\overline{Hist}(R_i) = \cup_j \overline{Hist}^j(R_i) = Hist(R_i) \cap Hist(R \ltimes S)$.*

e. A stage of the local semi-joins $R_i \ltimes S$ and $S_i \ltimes R$ computation:
Owing to the fact that, at the end of stage d., each processor i holds the fragment $\overline{Hist}(R_i) = Hist(R_i) \cap Hist(R \ltimes S)$, the semi-join $R_i \ltimes S$ (resp. $S_i \ltimes R$) can be computed, in parallel on each processor i, by a scan of the fragment R_i (resp. S_i) and consulting the histogram $\overline{Hist}(R_i)$ in time $O(\|R_i\|)$ (resp. $O(\|S_i\|)$). Thus, the total cost of this stage is: $Time_{\text{step.e}} = O\Big(\max_{i=1,..,p}(\|R_i\| + \|S_i\|)\Big)$.

The total cost of the two semi-joins $R \ltimes S$ and $S \ltimes R$ computation, is thus the sum of the above five stages:

$$Time_{sj} = O\Big(\min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + \max_{i=1,..,p} \|R_i\| + \min(g * |Hist(S)| + \|Hist(S)\|,\ g * \frac{|S|}{p} + \frac{\|S\|}{p}) + \max_{i=1,..,p} \|S_i\| + l\Big).$$

■

Using the previous results, the following proposition gives an upper bound the global BSP cost for semi-join computation.

Proposition 3. *Let $R_{i=1,..,p}$ and $S_{i=1,..,p}$ be the fragments of R and S on p processors. The semi-joins $R \ltimes S$ and $S \ltimes R$ can be carried out in time:*

$$Time_{semi-joins} = O\Big(c_{i/o} * \max_{i=1,..,p}(|R_i| + |S_i|) + \max_{i=1,..,p} \|R_i\| + \max_{i=1,..,p} \|S_i\| + l + \min(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}) + \min(g * |Hist(S)| + \|Hist(S)\|,\ g * \frac{|S|}{p} + \frac{\|S\|}{p})\Big),$$

where c_i/o is the cost to read/write a page of data from disk.

Proof: Starting from the local fragments $R_{i=1,..,p}$ (resp. $S_{i=1,..,p}$) of relation R (resp. S), the local histograms $Hist(R_i)_{i=1,..,p}$ (resp. $Hist(S_i)_{i=1,..,p}$) can be created in parallel by a scan of R_i (resp. S_i) on processor i in time: $c_{i/o} * \max_{i=1,..,p} |R_i|$ (resp. $c_{i/o} * \max_{i=1,..,p} |S_i|$). Hence this step costs:

$$Time_{sj_a} = O\Big(c_{i/o} * \max_{i=1,..,p}(|R_i| + |S_i|)\Big),$$

At the end of this step, each processor i holds $Hist(R_i)$ and $Hist(S_i)$.

By using Proposition 2 and by using an appropriate hashing function (see remark 1), the two semi-joins, $R \ltimes S$ and $S \ltimes R$ are computed in time:

$$Time_{sj_b} = O\Big(\min\Big(g * |Hist(R)| + \|Hist(R)\|,\ g * \frac{|R|}{p} + \frac{\|R\|}{p}\Big) + \max_{i=1,..,p} \|R_i\| + \min\Big(g * |Hist(S)| + \|Hist(S)\|,\ g * \frac{|S|}{p} + \frac{\|S\|}{p}\Big) + \max_{i=1,..,p} \|S_i\| + l \Big).$$

The semi-joins computation has the sum of the above two costs, which completes the proof of the proposition.

■

Remark 3. *Sequential semi-joins $R \ltimes S$ and $S \ltimes S$ require at least the time to scan input relations R and S: $bound_{inf_1} = \Omega\Big(c_{i/o} * |R| + \|R\| + c_{i/o} * |S| + \|S\|\Big)$. Parallel processing using p processors requires therefore $bound_{inf_p} = \frac{1}{p} * bound_{inf_1}$.*

We recall that, in all existing parallel semi-join algorithms, semi-joins are performed in two phases: a redistribution phase (generally based on hashing functions) according to the values of the semi-join attribute of input relations and then a sequential scan of local relations. In these algorithms, the redistribution phase is very sensitive to data skew [2, 3, 1].

Using our approach, the semi-join computation requires only the distribution of histograms which make it insensitive to data skew. The complexity of our algorithm is asymptotically optimal owing to the fact that all the terms of the global BSP cost are of the same order of those of $bound_{inf_p}$.

3 Conclusion

In this paper, we have introduced the first parallel semi-join algorithm allowing to reduce the communication costs to the minimum owing to the fact that only histograms are redistributed across the network and the sizes of these histograms are very small compared to the sizes of input relations.

Using the BSP cost model, the algorithm is proved to have an asymptotic optimal complexity while guaranteeing perfect balancing properties even in the presence of highly skewed data. This algorithm can be used efficiently to optimize the execution time of complex queries by reducing the amount of input data for each sub-query and the communication costs.

Future work will consist in studying how histograms could be used to generate efficient query execution plans for complex queries. We also plan to study how these histograms could help to optimize resource allocation in distributed architectures.

References

1. M. Bamha and M. Exbrayat. Pipelining a skew-insensitive parallel join algorithm. *Parallel Processing Letters*, Volume 3, Number 3, pages 317–328, 2003.

2. M. Bamha and G. Hains. A skew insensitive algorithm for join and multi-join operation on Shared Nothing machines. In *the 11th International Conference on Database and Expert Systems Applications DEXA'2000*, volume 1873, 2000.
3. M. Bamha and G. Hains. A frequency adaptive join algorithm for Shared Nothing machines. *Journal of Parallel and Distributed Computing Practices (PDCP), Volume 3, Number 3, pages 333-345*, September 1999.
4. J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, April 1979.
5. D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri. Practical Skew Handling in Parallel Joins. In *Proceedings of the 18th VLDB Conference*, pages 27–40, Vancouver, British Columbia, Canada, 1992.
6. K. A. Hua and C. Lee. Handling data skew in multiprocessor database computers using partition tuning. *Proc. of the 17th International Conference on Very Large Data Bases*, pages 525–535, Barcelona, Catalonia, Spain, 1991.
7. M. Kitsuregawa and Y. Ogawa. Bucket spreading parallel hash: A new, robust, parallel hash join method for skew in the super database computer (SDC). *16th International Conference on Very Large Data Bases*, pages 210–221, 1990.
8. M. Seetha and P. S. Yu. Effectiveness of parallel joins. *IEEE, Transactions on Knowledge and Data Enginneerings*, 2(4):410–424, December 1990.
9. D. B. Skillicorn, J. M. D. Hill, and W. F. McColl. Questions and Answers about BSP. *Scientific Programming*, 6(3):249–274, 1997.
10. Konrad Stocker, Donald Kossmann, Reinhard Braumandl, and Alfons Kemper. Integrating semi-join-reducers into state-of-the-art query processors. In *Proceedings of the 17th International Conference on Data Engineering*, pages 575 – 584, 2001.
11. Joel L. Wolf, Daniel M. Dias, Philip S. Yu, and John Turek. New algorithms for parallelizing relational database joins in the presence of data skew. *IEEE Transactions on Knowledge and Data Engineering*, 6(6):990–997, 1994.

Two Fundamental Concepts in Skeletal Parallel Programming

Anne Benoit and Murray Cole

School of Informatics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK
{abenoit1, mic}@inf.ed.ac.uk
http://homepages.inf.ed.ac.uk/mic/Skeletons

Abstract. We define the concepts of *nesting mode* and *interaction mode* as they arise in the description of skeletal parallel programming systems. We suggest that these new concepts encapsulate fundamental design issues and may play a useful role in defining and distinguishing between the capabilities of competing systems. We present the decisions taken in our own Edinburgh Skeleton Library *eSkel*, and review the approaches chosen by a selection of other skeleton libraries.

1 Introduction

The skeletal approach to parallel programming is well documented in the research literature (see [1, 2, 3, 4] for surveys). It observes that many parallel algorithms can be characterised and classified by their adherence to one or more of a number of generic patterns of computation and interaction. For instance, a variety of applications in image and signal processing are naturally expressed as process pipelines, with parallelism both between pipeline stages, and within each stage by replication and/or more traditional data parallelism [5].

Skeletal programming proposes that such patterns be abstracted and provided as a programmer's toolkit, with specifications which transcend architectural variations but implementations which recognise these to enhance performance. This level of abstraction makes it easier for the disciplined programmer to experiment with a variety of parallel structurings for a given application, by enabling a clean separation between structural aspects and the application specific details. Meanwhile, the explicit structural information provided by the use of skeletons enables static and dynamic optimisations of the underlying implementation.

In the *eSkel* (Edinburgh Skeleton Library) project [6, 7, 8], motivated by our observations [2] on previous attempts to implement these ideas, we have begun to define a generic set of skeletons as a library of C functions on top of MPI. This work has caused us to identify two issues which we believe to be fundamental to the definition of any skeletal parallel programming system, but which have been addressed only implicitly in previous work.

The main contribution of this paper, in Section 2, is to present these underlying concepts explicitly. Section 3 outlines the approach we take in *eSkel*,

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 764–771, 2005.
© Springer-Verlag Berlin Heidelberg 2005

comparing this in Section 4 with the main characteristics of a number of previous projects. Our conclusions are in Section 5.

2 Fundamental Concepts

The concepts highlighted by this paper may usefully be labelled *nesting mode* and *interaction mode*. In order to present these clearly and concisely, we begin by introducing some generic terminology for other important aspects of skeletal programming. We will then introduce the new concepts. In doing so, we will provide simple examples, which involve multiple uses of only one actual skeleton, the pipeline. This simplification allows us to focus precisely on the underlying issues, which might be obscured by the use of further skeletons and real applications.

2.1 Skeleton Terminology

The purpose of any skeleton, irrespective of the language framework within which it is embedded, is to abstract a pattern of *activities* and their *interactions* (where the activities might be processes or threads, and the interactions might be message based communications or exchanges through shared variables). Activities might themselves be internally parallel.

It is helpful to further distinguish *internal* and *external* interactions. *Internal* interactions occur between two or more activities, whereas *external interactions* occur between activities and the enclosing context. For example, in a pipeline, an interaction between two stages is internal, but an interaction between the first stage and the source of pipeline inputs (which might be a data buffer, or some other thread or process, or even another skeleton) is external.

The definition of a particular skeleton constrains the ways in which its constituent activities may interact. In general such constraints have two aspects. For a given activity, *spatial constraints* determine the activities with which it *may* interact and the directions these interactions may take. We will call these the *partner activities*. For example, in a pipeline, activities correspond to stages and for a given stage its partners are the preceding and succeeding stages (where these exist), with the data restricted to flow *from* the predecessor *to* the successor. Meanwhile, *temporal constraints* determine the allowable orderings of interactions between partners. For example, in a particularly strict form of pipeline, we may require that a stage interacts first with its predecessor then its successor, in strict alternation.

2.2 Fundamental Concept: Nesting Mode

Conceptually, skeletons may be nested in two ways. In the discussion that follows we will refer to *outer* and *inner* levels to indicate what might be respectively thought of as the "calling" and "called" skeleton instances in a nesting. In a multiple level nesting, all but the outermost and innermost skeleton instances will play both roles.

Transient Nesting. In a **transient nesting**, an activity may decide to invoke another skeleton in order to process some locally available data, or to perform some other self-contained computation. For example, a pipeline stage with several internal processes may receive an input datum, partition it into a collection of sub-data, and process these independently with an internal pipeline, before reconstituting a single outer level result datum from the collected inner level results. This is then transmitted to the subsequent outer level stage. Each invocation of the nested pipeline is initiated by the outer level activity, between, and so independently of, interactions at the outer level. A transient nesting is therefore really no more than a conventional function call, albeit one invoked concurrently by all members of the outer level activity.

Persistent Nesting. In contrast, in a **persistent nesting** an activity at the outer level handles a complete sequence of enclosing level interactions in a skeletally structured fashion. From the perspective of the inner skeleton, its external interactions become bound to interactions (which may be internal or external depending upon context) at the outer level. For example, a pipeline stage (at the outer level) could choose to invoke a persistent pipeline as its inner level. Each request for a datum by the first stage of the inner level would be (conceptually at least) mapped to a request for a datum by the enclosing activity (ie the outer level pipeline stage). The inner skeleton invocation persists for the duration of the outer skeleton invocation. In particular it persists across outer level interactions, and indeed *subsumes responsibility* for dealing with these.

Example & Discussion. We now illustrate the nesting modes with a simple example. The example is deliberately artificial, since our intention is to contrast the essence of the modes without application specific clutter. For the same reason, we use only the pipeline skeleton, since we wish to focus on the interaction and nesting mechanisms, rather than the particular patterns encapsulated.

Consider a four stage pipeline, in which each stage (outer level activity) is assigned three processors. The pipeline processes 100 inputs, where each input is a block of 10 integers. Thus, each outer level activity performs 100 input and 100 output interactions. Now consider some possible nested structures for the second stage.

We could choose to design a second stage as follows. Upon receipt of each new input, the datum is decomposed into a sequence of 10 individual integers, which are then passed through a three stage inner level pipeline formed by the processors responsible for this outer level stage. Having processed the ten integers, the inner pipeline is dissolved (its call terminates) and the ten results are reconstituted into a single outer level datum, for forwarding to the third stage at the outer level. The process repeats with the next outer level datum. For each invocation of the inner level pipeline, each processor within the second outer level stage processes a stream of 10 inputs in its role as an inner level pipeline stage. Since there will be 100 such invocations, each processor processes 1000 inputs (each a single integer) in total. This is a transient nesting.

In a different application but with the same data types and structure at the outer level, the second outer level stage might choose to use its three processors

as a nested persistent pipeline. In this case, a request for a new datum by the first stage at the inner level corresponds directly to a request for an input datum by the second stage at the outer level. The nested skeleton is invoked once for the entire computation. Now each of the three second (outer) stage processors processes 100 inputs (each a block of 10 integers) *in total* during its role as an inner level pipeline stage. It is interesting to note that in this artificial example, the structure is operationally equivalent to that which could have been obtained by expressing the structure as a single level, six (NB not seven) stage pipeline. This is entirely natural, and serves to highlight the distinction between persistent and transient nesting (since this equivalence does not hold in the transient case).

2.3 Fundamental Concept: Interaction Mode

The concept of *interaction mode* concerns the extent to which a skeleton constrains temporal, as well as spatial interaction. In general, it is reasonable to consider that this property may vary from one invocation of the same skeleton to another, and even between the constituent activities of one invocation. Thus, interaction mode is a property of an activity.

Implicit Interactions. An activity using **implicit interaction mode** makes interactions which are constrained both spatially and temporally by the skeleton.

Explicit Interactions. An activity using **explicit interaction mode** makes interactions which are constrained spatially by the skeleton but which are unconstrained temporally, being triggered explicitly instead by actions in the activity code.

Examples & Discussion. Consider a stage in an image processing pipeline, which is producing one transformed output image for every input image. We might think of this as the "normal" behaviour for a pipeline. It can be completely encapsulated by the constraints of the *implicit* mode for a pipeline stage activity. Another pipeline might contain a stage activity which is behaving as a filter: some inputs are accepted and passed on (perhaps after processing) to the subsequent stage, while others are removed from the stream. In this case some explicit action is required of the activity to trigger each input interaction. Consequently the activity has *explicit* interaction mode. Finally, imagine a stage which is generating several output interactions for each input interaction (or indeed without any input interactions at all, as occurs for example in the "generator" stage of the well known "prime sieve" pipeline). Again, some *explicit* trigger would be required to indicate availability of a new output. In general, we can imagine that a fully flexible skeleton should allow the programmer to specify the interaction mode for each activity.

3 Nesting and Interaction in *eSkel*

The *eSkel* library [8] provides the C/MPI programmer with a set of skeletal collective operations. We now explain how version 2 of the library addresses the

fundamental issues discussed in the previous section. As before, to avoid clutter we will not discuss eSkel's specific collection of skeletons, since these are orthogonal to the handling of modes. Activities in *eSkel* are defined as C functions, to be called in SPMD style by all processes participating in the activity, and passed to the skeleton call as function pointers.

Nesting Mode. *eSkel* allows both transient and persistent nesting in order to maximize flexibility. In concrete terms, the nesting mode is actually expressed through what *eSkel* calls the "data mode" parameter to a skeleton call. This determines whether the data which will be processed by the skeleton invocation is present in a (possibly distributed) buffer, provided by the calling processes as a further parameter to the call (so called "buffer mode" data) or that it will flow into the skeleton from the activities (and indirectly, the buffer) of some enclosing skeleton call (so called "stream mode" data). Thus, a skeleton called in buffer mode effectively carries or creates its own data, and so is *transiently* nested with respect to any enclosing skeleton call. In contrast, a skeleton called with stream data mode is by necessity (and implication) *persistently* nested. As an aside, we note that the outermost skeleton call in any *eSkel* program must have buffer mode data (because there is nowhere else for data to come from), and so is effectively transiently nested within any enclosing (non-skeletal) code.

Interaction Mode. In *eSkel*, a distinct interaction mode is associated with each activity in a skeleton instantiation by setting corresponding parameters to the skeleton call. For non-nested skeletons this involves a simple choice between the two standard modes, *implicit* or *explicit*, as discussed above. In the case of nested skeletons, we allow a further level of flexibility, in that a skeleton call may devolve its choice of interaction mode for any of its activities to be determined by a further skeleton, nested within that activity's code. This is achieved by specifying that the outer level activity has *devolved* activity mode. This process can be repeated arbitrarily many times within a hierarchy of nestings, to be ultimately resolved by the "leaf" activities, which must have one of the two fundamental interaction modes.

An implicit mode activity has no control over its interactions, so is described in *eSkel* as a function from input data items to output data items. In general, a single interaction may involve several inputs and/or several outputs, to various partner activities (for example, consider the "stencil" interaction at the heart of a typical two dimensional iterative relaxation algorithm). The implicit mode activity function is invoked every time a new collection of inputs arrives (under the control of the library implementation of the skeleton, which is taking care of the interactions). The semantics of the interactions are specified in terms of **eDM**, the *eSkel Data Model*. The unit of transfer during the interactions is an *eDM molecule*, which consists of a collection of *eDM atoms* (each containing the data relating to a distinct partner activity). The type of the data is defined using standard MPI datatypes. Full details of the eSkel data model are presented in [2].

By contrast, an activity with explicit interaction mode is invoked only once per skeleton call. Its interactions are triggered directly in the activity code, by

calling the functions `Take` and `Give`. These have a generic interface, but semantics defined for each skeleton (embodying the skeleton's spatial constraints). For example, calling `Give` in an explicit mode pipeline stage indicates that the data given as a parameter is ready for transfer to the successor stage.

4 The Fundamental Concepts in Other Skeleton Libraries

We now briefly introduce a selection of existing skeleton languages and libraries and consider their handling of nesting and interaction. The results are summarized in the following table, which should be read in conjunction with the discussion below, particularly for the cases annotated with *.

Library	Interaction mode		Nesting mode	
	Explicit	Implicit	Persistent	Transient
P3L	No	Yes	Yes	No
Lithium	No	Yes	Yes	No*
ASSIST	Yes	Yes	No*	No*
Kuchen	Yes*	Yes	Yes	Yes*
Eden	Yes	Yes*	Yes*	Yes*

P3L. The Pisa Parallel Programming Language (P3L) [3, 9] was designed to help the design of parallel applications by providing skeletons as new language constructs (rather than a library) which co-ordinate fragments of sequential code written in C. The **Anacleto** [10] compiler for P3L generates C and MPI. P3L inspired industrial collaboration, leading to **SkIE** [11], a heterogeneous environment for high performance computing applications. SkIE includes a graphical editor, allowing fast and intuitive design of the parallel components of an application. Interaction within P3L skeletons is via streams of data, which are interfaced to the sequential code through parameter lists and manipulated through ordinary C types and operations. No choice is given for the interaction mode - all interactions are implicitly defined by the skeleton. All nesting of skeletons is persistent, being defined within the P3L layer, which is syntactically separated from the sequential code defining the activities. Transient nesting is therefore not possible.

P3L-based Libraries. One of the first attempts to provide a skeleton programming environment via a plain C library was achieved with **SKElib** [12], which was based on the P3L conceptual (but not syntactic) model. Similarly, **Lithium** [13] is a more recently developed library which takes a corresponding approach on top of Java. These libraries inherit their interaction modes from P3L. In Lithium, skeletons and skeleton nestings are constructed as objects which are then executed by invoking a `parDo` method. This form of nesting is persistent. The concept of transient nesting is not explicitly addressed, and

while not syntactically forbidden, it appears that the current implementation does not support it.

ASSIST. More recently, the programming environment ASSIST [14] (A Software development System based upon Integrated Skeleton Technology) is being developed in Pisa. It allows more flexibility than the original P3L approaches. It proposes a generic skeleton, *parmod*, which has few constraints but includes the classical P3L style skeletons. The interactions can be either implicit and constrained by a skeleton, or can be explicitly defined by the user. No nesting of skeletons is allowed - complex structures must be expressed directly through the flexibility afforded by *parmod*.

Kuchen's Skeleton Library. Herbert Kuchen has proposed a skeleton library [15] on top of C++. This library proposes polymorphic higher-order functions which are efficiently implemented in parallel. It offers data parallel and task parallel skeletons, and a two level model in which data parallel skeletons may be invoked from within task parallel skeleton nests. Most interactions have implicit mode, but explicit mode is also supported for task parallel skeletons. Nesting of task parallel skeletons is persistent since, like Lithium, the task parallel process topology is fixed at construction. The data parallel skeletons allow more flexibility with transient nesting performed via direct calls to the functions.

Eden. The parallel functional language Eden [16] provides a model which is explicit about processes and their incoming and outgoing data, but abstracts from the detailed transfers and synchronisation. An Eden program defines a system of processes exchanging data on communication channels which are visible to the programmer as lazy lists. Explicit interactions are invoked by explicit manipulations of these lists. A number of skeletons have been defined on top of the Eden model [4, 17]. The examples presented leave the programmer with explicit control of interaction, but it would be simple to define further skeletons which abstract to implicit mode. Nesting issues are not addressed but it is clear that transient nesting can be achieved by calling a skeleton function at any time in the inner code of a skeleton, while persistent nesting can be achieved by appropriate use of Eden's process composition.

5 Summary and Conclusions

We have introduced the concepts of *nesting mode* and *interaction mode* as aspects of the design of skeletal parallel programming frameworks. We have examined several existing skeletal systems and have observed that design decisions related to these concepts are often made implicitly, and perhaps even as side-effects of other design decisions. We argue that these issues should be addressed explicitly in a design, and in its specification. We believe that the conceptual basis established here will help future designers to clarify and justify their decisions. We are attempting to follow this philosophy in the ongoing design and development of the *eSkel* library.

References

1. Cole, M.: Algorithmic Skeletons: Structured Management of Parallel Computation. MIT Press & Pitman, ISBN 0-262-53086-4 (1989)
2. Cole, M.: Bringing Skeletons out of the Closet: A Pragmatic Manifesto for Skeletal Parallel Programming. Parallel Computing **30** (2004) 389–406
3. Pelagatti, S.: Structured Development of Parallel Programs. Taylor & Francis, London (1998)
4. Rabhi, F., Gorlatch, S., eds.: Patterns and Skeletons for Parallel and Distributed Computing. Springer Verlag, ISBN 1-85233-506-8 (2003)
5. Subhlok, J., O'Hallaron, D., Gross, T., Dinda, P., Webb, J.: Communication and memory requirements as the basis for mapping task and data parallel programs. In: Proceedings of Supercomputing '94, Washington, DC (1994) 330–339
6. Cole, M.: eSkel: The **e**dinburgh **Ske**leton **l**ibrary. Tutorial Introduction. Internal Paper, School of Informatics, University of Edinburgh (2002)
7. Cole, M.: eSkel: The **e**dinburgh **Ske**leton **l**ibrary Version 2.0 – Draft API reference manual. Internal Paper, School of Informatics, University of Edinburgh (2003)
8. Benoit, A., Cole, M.: eSkel's web page. (2005) `http://homepages.inf.ed.ac.uk/mic/eSkel`.
9. Bacci, B., Danelutto, M., Orlando, S., Pelagatti, S., Vanneschi, M.: P3L: A structured high level programming language and its structured support. Concurrency: Practice and Experience **7** (1995) 225–255
10. Ciarpaglini, S., Danelutto, M., Folchi, L., Manconi, C., Pelagatti, S.: ANACLETO: a template-based P3L compiler. In: Proceedings of the PCW'97, Camberra,Australia (1997)
11. Bacci, B., Danelutto, M., Pelagatti, S., Vanneschi, M.: SkIE: a heterogeneous environment for HPC applications. Parallel Computing **25** (1999) 1827–1852
12. Danelutto, M., Stigliani, M.: SKElib: parallel programming with skeletons in C. In: Proceedings of EuroPar2000. Number 1900 in LNCS, Springer-Verlag (2000)
13. Aldinucci, M., Danelutto, M., Teti, P.: An advanced environment supporting structured parallel programming in Java. Future Generation Computer Systems **19** (2003) 611–626
14. Aldinucci, M., Coppola, M., Danelutto, M., Vanneschi, M., Zoccolo, C.: ASSIST as a Research Framework for High-performance Grid Programming Environments. In Cunha, J.C., Rana, O.F., eds.: Grid Computing: Software environments and Tools, Springer Verlag (2004)
15. Kuchen, H.: A skeleton library. In: Proceedings of the 8th International Euro-Par Conference on Parallel Processing, Springer-Verlag (2002) 620–629
16. Breitinger, S., Loogen, R., Ortega-Mallén, Y., Peña, R.: Eden: Language Definition and Operational Semantics. Technical Report 10, Philipps-University of Marburg (1996)
17. Galán, L.A., Pareja, C., Pena, R.: Functional Skeletons Generate Process Topologies in Eden. In: Proc. of the 8th International Symposium on Programming Languages: Implementations, Logics, and Programs, Springer-Verlag (1996) 289–303

A Formal Framework for Orthogonal Data and Control Parallelism Handling

Sonia Campa

Dept. Computer Science – University of Pisa,
Viale Buonarroti 2, I-56125 PISA – Italy
campa@di.unipi.it

Abstract. We propose a semantic framework for parallel programming based on the orthogonalization of data access and control concerns by means of set of abstraction mechanisms. Such mechanisms regard the description of how data has to be accessed, the description of how data has to be computed and the description of how coupling data accesses and patterns of control. Each description is represented by an abstraction mechanism formalized through a formal semantics. The set of semantics specifications defines a method to investigate the structure of the whole application. We demonstrate how this semantics provides a formal, provable method to statically or dynamically evaluate the overall performance of the application and, eventually, apply optimization rules.

1 Introduction

From the user's point of view, one of the hardest trouble in writing parallel programs is represented by the need of expressing both data and control parallel concerns at the same time. Expressing data parallelism leads to the definition of how input data have to be accessed (e.g. sequentially or in parallel); expressing control parallelism regards the description of the functional structure of the computation, hence how each accessed item has to be processed in order to get the final result. These concerns are intrinsically orthogonal and writing a parallel application today still means keeping them together in a mixture of data accessing and control code.

Since the structured parallel programming model [8] has been assessed, many works have been proposed with the aim of supporting a higher level of abstraction in programming parallel applications and, at the same time, to get the best performance. Nevertheless, nearby the extreme cases of data-driven [1, 5] and control-driven [3, 13] approaches, the most works define a two-tier architecture model in which data accesses have to be heavily detailed on top of control pattern abstractions [6, 9] or vice versa [10, 11, 12, 14]. As a consequence, data (control) intensive applications written through control-centric (data-centric) artifacts could expose a model of parallelism difficult to describe and manipulate and, in practice, a performance that is difficult to predict or to optimize.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 772–780, 2005.
© Springer-Verlag Berlin Heidelberg 2005

This work aims at defining a new parallel programming model semantics that exposes a set of abstractions to express data and control concerns in an independent manner. The semantics provides some primitives to couple data and control expressions giving them a precise parallel behavior and to construct the whole application. Moreover, as further goal of our approach, giving data and control semantics in an independent but composite manner will be the baseline for the definition of a set of transformation rules able to evaluate and optimize the overall performance at compile and/or at run time.

Section 2 gives an overall introduction to the model we are working on. Section 3 details the semantic formulation of the outlined model. Section 4 shows some examples of formalization and how applying some preliminary optimization rules. The section is closed with a brief outline of a first Java implementation of the semantic framework; Section 6 compares related works and Section 7 summarizes conclusions and future work.

2 The Model

The basic idea of our approach is to provide the programmer with some primitives to describe separately the two main concerns of a parallel program: data accesses and patterns of control. In particular, patterns of control are represented by compoundable *control primitives* describing the functional structure of the computation. As consequence, the application results in a graph in which the nodes represent sequential or parallel modules and the arcs represent functional dependencies among modules. Such a graph is formally described by a set of semantic operators and the framework provides an optimized implementation of the related language primitives.

On the other hand, parallel or sequential accesses on data are described by means of three mechanisms: *abstract data types*, *views* and *iterators*. An abstract data type is a representation of the raw input data that abstracts from the actual implementation and/or distribution of data. Moreover, the model provides different kind of typed views that can be declared on top of an abstract data type. Declaring a view of type v on an abstract data type a means declaring a logical structure of the raw data, so that the user can think of (and program) a as it was a v data structure, regardless its actual implementation. For example, an array view could be defined on top of a set of items evaluating it as a collection of singleton or a collection of data blocks. Further, the programmer could apply different kind of views on the same abstract data type in order to take advantage of different logics and operators to manage its data. The logical separation of input raw data and views will allow the framework to evaluate and optimize the actual implementation of an input data set and the abstract mechanisms used to manipulate it.

Each view provides a set of typed iterators on its items. An iterator is an object exposing a set of operations to get items from the view, coherently with the view type. For example, an array view provides iterators to get singleton or

block of items, while a tree view provides iterators to get subtrees, children or siblings of a given node, and so on. The behavior of the iterator can be specialized in order to get items in a whole or sequentially.

Summarizing, the framework provides control primitives able to describe the functional structure of an application and instantiations of views and iterators on the abstract data types that are able to describe the access patterns to the input data. The glue between these two separated concerns is represented by a set of collective primitives that couple each selected control pattern with one or more iterators. The internal parallel behavior of each collective depends on the specific couple control pattern-iterator by which it is constructed. For example, let us consider an application that can be evaluated applying the same function to all the items belonging to an array view. The application involves only one pattern of control (it is called `apply-to-all` and applies a function to all the items of its input in parallel) and the user can decide if instantiating an array view's iterator that returns the items in a whole or one-by-one. In the first case, combining the iterator with the `apply-to-all` control pattern will exploit a plain data parallel behavior[1] , in the second case the combination acts as a plain task parallel behavior[2].

By using such collectives and mechanisms, a programmer has independently described the computational graph and the policies to access data, leaving to the framework the optimization of the actual implementation. The optimization improvements arise from transformation rules that are statically and/or dynamically applied by the framework through the associated algebra, as demonstrated in the next two sections.

3 Semantics

Our model is completely described by the tuple

$$\mathcal{M} =< \mathcal{A}, \mathcal{V}, \mathcal{I}, \mathcal{C} >$$

where $\mathcal{A}$ stands for a set of abstract data types representing the raw data types, $\mathcal{V}$ is the set of view types, $\mathcal{I}$ is the set of iterator types, and $\mathcal{C}$ is the set of collective primitives.

The view abstraction is built by a set of mapping functions that, given an abstract data type $\{s_1, \ldots, s_n\} \in \mathcal{A}$ as input, returns an object belonging to the set of view types $\mathcal{V}$. At the moment, the provided view types are `Array, Matrix, Graph, Tree` and `List`. For example, given $s = \{s_1, ..., s_n\} \in \mathcal{A}$, and $g \in \mathbb{N}$ as a grain value, the `Array(s,g)` type is constructed by the mapping function

$$\mathcal{A} \times \mathbb{N} \rightarrow \texttt{Array}(\mathtt{s}, \mathtt{g}) = [(s_1, \ldots, s_g), (s_{g+1}, \ldots, s_{2g}), \ldots, (s_{size(s)-g}, \ldots, s_n)]$$

[1] Pattern called `map` in the skeleton community's jargon.
[2] Pattern called `farm` in the skeleton community's jargon.

Each view is a factory for a set of iterators on its structure. Generally speaking, an iterator is a couple (v, p) where $v \in V$ is the (sub)view on which it applies and $p \in \mathbb{N}$ is a pointer to the next accessible item of its range. Thus, an iterator is given by the function

$$\mathcal{V} \rightarrow \mathcal{V} \times \mathbb{N}$$

that constructs the couple $(v, p) \in \mathcal{I} = (\mathcal{V} \times \mathbb{N})$ on a given view, according to the factory rules of the view type. Tab. 1 shows some operations to handle an iterator object. Type $\mathcal{O}$ represents a super type for objects. The operator *curr* returns the current item which the pointer p points to. If the iterator has a parallel behavior, then p "points" in parallel to all the items so that they are taken as a whole; otherwise p points also to the current item and the operator returns it as a singleton.

Table 1. Sample operators on iterators

$curr : \mathcal{I} \rightarrow \mathcal{O}$	$curr((v,p)) = v_p$
$skip : \mathcal{I} \rightarrow \mathcal{I}$	$skip((v,p)) = (v, p+1)$
$hasNext : \mathcal{I} \rightarrow Bool$	$hasNext((v,p)) = (p \leq \#v)$
$\vdots$	$\vdots$

Using $\mathcal{A}$, $\mathcal{V}$, $\mathcal{I}$ objects and its operators allows to describe access patterns to the input data. On the other hand, it is possible to describe control patterns by means of a set of control primitives. Typically, a control primitive is a function $\mathcal{C} : \mathcal{I} \times (\mathcal{C} \cup Function) \rightarrow \mathcal{V}$ that gets a sequential function (its type is called *Function*) or a nested control primitive and a set of values (i.e. the ones given by one (or more) coupled iterator(s)) and returns the view obtained by applying the second argument to the item set. The parallel access behavior by which this item set will be accessed is encapsulated by the iterator instance given as input: this means that the programmer can concentrate on the structural aspects of his computational graph, regardless how to deal with data parallel concerns.

In our semantic model, operators on iterators and collectives (that is the control pattern concerns) are evaluated by means of inference rules that define how each of them has to be transformed by the evaluation relation $\rightarrow$.

Let us give some example of how this relation works and on which operators.

Since an **iterator** is represented as a couple $(\overline{x}, p)$ where $\overline{x} = [x_1, ..., x_n]$, the *curr* operator is in charge of returning the p-th element (*SeqIterator* type) or all the elements as a whole (*ParIterator* type). The *curr* operator is evaluated by the following rule:

$$\frac{it = ([x_1, \ldots, x_n], p) : SeqIterator}{curr(it) \rightarrow^{curr} x_p} \quad \frac{it = (x, p) : ParIterator}{curr(it) \rightarrow^{curr} \overline{x}} \tag{1}$$

The operator that describes the parallel evaluation of two functions f and g is given by $f \parallel g$ (**par** operator). For example, two of the rules leading the evaluation of such an operator are the ones assessing the evaluation order:

$$a.\frac{g(x) \rightarrow y' \wedge f(x) \rightarrow x'}{h(g(x)) \parallel h'(f(x)) \rightarrow h(y') \parallel h'(x')} \qquad b.\frac{g(y) \rightarrow y' \wedge f(x) \rightarrow x'}{f(x) \parallel g(y) \rightarrow^{par} [x', y']} \quad (2)$$

The evaluation of `par` is strict i.e. before evaluating it, all its arguments have to be completely evaluated. Hence, in case of nested functions (rule 2-a), the inner functions have to be evaluated, first. Finally, the evaluation of `par` produces a new view by appending all the parallel results in a new data set(rule 2-b).

Let f be a function to be applied to all the items provided by an iterator $it = ([x_1, \ldots, x_n], 1)$ and let $v = [f(x_1), \ldots, f(x_n)]$ be the functional result it waits for. We can compute this transformation by means of an operator that gets the couple (it, f) and, depending on the type of it (parallel or sequential), applies f to all the *curr*'s of it. The inference rules describing such a "spreading" operator (called **spread**) are given by:

$$\frac{curr(it) \rightarrow [x_i | \forall i = 1, \ldots, size(it)]}{f \ominus it \rightarrow^{spr} \|_{i=1,..,size(it)}\ f(x_i)} \qquad \frac{curr(it) \rightarrow x_p}{f \ominus it \rightarrow^{spr}\ f(x_p); f \ominus skip(it)} \quad (3)$$

As it can be seen, the evaluation of the spreading operator $\ominus$ depends on the evaluation of *curr* thus, in chain, on the type of it. If it is sequential, f is applied sequentially to all the element of it by invoking $\ominus$ recursively on a successive state of it each time. Such state is provided by the `skip` operator whose evaluation rules are very intuitive. The operator "`;`" stands for the sequential evaluation of its arguments. If it has a parallel behavior, *curr* returns all the element as a whole and they need to be evaluated by the `par` operator.

Having the set of operators given above, we are now able to describe the **apply-to-all** (α) control pattern that in parallel applies f to each element of an input dataset:

$$\frac{hasNext(it) \rightarrow true}{\alpha(f, it) \rightarrow f \ominus it \parallel \alpha(f, skip(it))} \qquad \frac{hasNext(it) \rightarrow false}{\alpha(f, it) \rightarrow []} \quad (4)$$

The α operator depends on the more generic spread operator: if it is a parallel iterator, $\ominus$ will evaluate f on all the elements of it *in a whole* and the recursive call will stop the evaluation. This pattern exploits a plain data parallel behavior.

On the other hand, if it is a sequential iterator, $\ominus$ evaluates on the current singleton of it while α is invoked recursively and in parallel on the next state of it. The evaluation results in applying f to all the elements of it in but accessing the elements sequentially: this kind of pattern exploits a plain task parallel behavior.

A **pipeline** of functions can be modeled by an array or list view on the set of stages to be computed $v = [f_1, \ldots, f_n]$ and by a sequential iterator on such view, say $it_f = (v, p)$. Let it_d be a sequential iterator on the given input data set view and let $it_f \oplus x = f_n(f_{n-1}(...(f_2(f_1(x)))...))$ be the function that evaluates the pipeline of functions given by it_f on a single value x. The operator $\oplus$ (so-called **chain** operator) is evaluated through the following inference rules:

$$\frac{hasNext(it_f) \rightarrow true}{it_f \oplus x \rightarrow^{chain}\ skip(it_f) \oplus curr(it_f)(x)} \qquad \frac{hasNext(it_f) \rightarrow false}{it_f \oplus x \rightarrow^{chain}\ x} \quad (5)$$

Now we can give the inference rule able to describe a pipeline computation represented by it_f on the whole input data set accessed through the iterator it_d:

$$\frac{hasNext(it) \rightarrow true}{Pipe(it_f, it_d) \rightarrow^{pipe} it_f \oplus curr(it_d) \parallel Pipe(it_f, skip(it_d))} \quad (6)$$

$$\frac{hasNext(it_d) \rightarrow false}{Pipe(it_f, it_d) \rightarrow^{pipe} []} \quad (7)$$

It has to be pointed out that all the operators given above, are fully described by a complete set of inference rules but here we have just given as much as we consider to be sufficient.

4 Examples of Programs Evaluation

In order to show the expressive power of our formalization, we will give a simple example. Let $it_f = (< f_1, f_2 >, 1)$, $it_d = (< x_1, \ldots, x_3 >, 1)$ and $it^{(p)}$ be the notation for iterators stressing that the current state of it points to the p-th element. Moreover, let $Pipe(it_f, it_d) =< f_2(f_1(x_1)), f_2(f_1(x_2)) >$ be the expression we want to evaluate. By applying our rules, the evaluation proceeds as follows:

$Pipe(it_f, it_d)$
$\rightarrow^{pipe} \{(6)\} it_f^{(1)} curr(1)(it_d) \parallel Pipe(it_f, skip^{(2)}(it_d))$
$\rightarrow parcurr(1)(it_d)x_1, skip(2)(it_d) it_d it_f x_1 \parallel Pipe(it_f, it_d)$
$\rightarrow^{par}$ {we have to reduce `par`'s arguments, first}
$skip^{(2)} \oplus (it_f)(curr^{(1)} it_f) \oplus (x_1) || (it_f curr^{(2)}(it_d) \parallel Pipe(it_f, skip^{(3)}(it_d)))$
$\rightarrow^{par} it_f \oplus f_1(x1) \parallel (it_f \oplus x_2) \parallel Pipe(it_f, it_d^{(3)})$
$\rightarrow^{par} \{f_1(x_1) y_1\} it_f^{(2)} \oplus y_1 \parallel (it_f \oplus x_2) || Pipe(it_f, it_d^{(3)})$
$\rightarrow^{par} skip^{(3)}(it_f) \oplus curr^{(2)}(y_1) \parallel skip^{(2)}(it_f) \oplus curr^{(1)}(it_f)(x2) \parallel it_f curr^{(3)}(it_d) \parallel Pipe(it_f, skip^{(4)}(it_d))$
$\rightarrow^{par}$ {we have to reduce `par`'s arguments, first}
$it_f^{(3)} \oplus f_2(y_1) \parallel it_f^{(2)} \oplus f_1(x_2) \parallel it_f \oplus x_3 \parallel Pipe(it_f, it_d^{(4)})$
$\rightarrow^{par} \{f_2(y_1) \rightarrow y_1' \wedge f_1(x_2) \rightarrow y_2\} it_f^{(3)} y_1' \parallel it_f^{(2)} \oplus y_2 \parallel it_f \oplus x_3 \parallel Pipe(it_f, it_d^{(4)})$
$\rightarrow^{par} \{ hasNext(it_f) \rightarrow false \wedge hasNext(it_d) \rightarrow false\}$
$y_1' :: skip^{(3)}(it_f) \oplus curr^{(2)}(y_2) \parallel (skip^{(2)}(it_f) \oplus curr^{(1)} it_f(x_3)) \parallel []$
$\rightarrow^{par} y_1' :: it_f^{(3)} \oplus f_2(y_2) \parallel it_f^{(2)} \oplus f_1(x_3) \parallel []$
$\rightarrow^{par} \{f_1(x_3) \rightarrow y3 \wedge f2(y2) \rightarrow y_2'\} y_1' :: it_f^{(3)} \oplus y_2' \parallel it_f \oplus y_3 \parallel []$
$\rightarrow^{par} \{hasNext(it_f) \rightarrow false\} < y_1', y_2' >:: skip^{(3)}(it_f) \oplus curr^{(2)}(y3)$
$\rightarrow^{cons} < y_1', y_2' >:: it_f^{(3)} \oplus f_2(y_3) \rightarrow^{cons} \{f_2(y_3) \rightarrow y_3' \} < y_1', y_2' >:: it_f^{(3)} \oplus y_3'$
$\rightarrow^{cons} \{ hasNext(it_f) \rightarrow false\} < y_1', y_2', y_3' >$

In the previous transformation the *cons* (::) operator has been used for appending elements in the final list of results.

5 Transformation Rules

Once an user application has been written by means of our formalism, we could be interested in finding a semantic expression that is functionally equivalent to the user one but that exploits a better overall performance. As an example, we will refer to a classical rule involving the composition of functions presented in [4] and stating that given two functions, f and g, $\alpha((f;g),it) \leftrightarrow (\alpha(f,it);\alpha(g,it'))$ holds. In [2] it has been proved that the left-side application is more efficient than the right-side one. In the following we will show how the two-side reduction can be proved through our rules and as a consequence, how a more efficient program for the given expression can be statically found.

By applying the rules given above step by step and starting from $\alpha((f;g),it)$ we can obtain the following left-to-right transformation:

$$
\begin{aligned}
&\alpha((f;g),it)\\
&\rightarrow \alpha(f;g) \ominus it \parallel \alpha((f;g), skip(it))\\
&\rightarrow^{*} (f;g)(x_1) \parallel (f;g)(x_2) \parallel \cdots \parallel (f;g)(x_n)\\
&= g(f(x_1)) \parallel g(f(x_2)) \parallel \cdots \parallel g(f(x_n))\\
&= \{\text{let } it \text{ iterator on } [< f(x_1), \ldots, f(x_n) >]\}\ g \ominus it \parallel \alpha(g,it') = \alpha(g,it')\\
&= \{\text{let } it \text{ iterator on } [\alpha(f,it)]\}\ \alpha(f,it);(g,it').
\end{aligned}
$$

On the other hand, the right-to-left side of the transformation can be easily proved by applying the same steps in an inverse order.

The semantics provided so far allows us to describe the behavior of parallel programs in a uniform manner, simply involving iterator and control pattern concepts. The advantage gained immediately is comparing program behaviors and structures: we can statically define transformations between programs that are semantically equivalent but that exhibit different performance when implemented on the target architecture. Since these transformations are well described within the semantic framework, they can be easily implemented, evaluated and applied without the programmer intervention.

Just to prove the feasibility of our approach, we have developed a first Java **??** practical environment [7] implementing the semantic framework. At the moment, it offers `Array`, `Graph`, `Tree`, `List` and `Matrix` view abstractions to be treated. Some preliminary experimental results based on matrix multiplication obtained on a single versus a dual processors architecture have shown a good scalability trend with an efficiency near to 97%.

6 Related Works

The idea of using the "iterator" concept as a means of "ranging" data is not new in the field of parallel programming. In the STAPL library [14], for example, iterators are called *prange*, they work inside distributed data structures or ("containers") and they represent pieces of data on which a given node should compute. Although the approach seems the same, with respect to STAPL our

abstract mechanisms are quite different. In particular we use views instead of containers for organizing input data and different views can be applied to the same input data. Moreover, in STAPL a semantic basis leading to a static performance analysis is completely missing.

7 Conclusions and Future Work

We have outlined a formal basis for expressing in an orthogonal, independent manner data and control concerns in a parallel program by means of separated but compoundable abstraction mechanisms and operators. With respect to our previous work in which we proved the feasibility of this approach based on iterators and primitives by scratching a first implementation framework, the main focus of this work has been the introduction of a semantics associated to both the basic abstractions and operators which leads to the formal definition and evaluation of transformation rules. We have shown how such transformation can be done in order to optimize the parallel behavior of the whole application.

Future work will address the extension of the semantics by new control patterns (i.e. the irregular ones as *D&C* or broadcast patterns) and new transformation rules related to the set of the new given operators. Moreover, we are working on a costs model associated with the transformation rules through which predicting how much each transformation costs and, as a consequence, which one of two functionally equivalent semantic expressions is cheaper, i.e. more efficient.

Also, we will support such extensions into our Java prototype.

References

1. Sudhir Ahuja, Nicholas Carriero, and David Gelernter. Linda and friends. *Computer*, 19(8):26–34, August 1986.
2. M. Aldinucci and M. Danelutto. An operational semantics for skeletons. In *Proceedings PARCO'2003*, 2003. to appear.
3. F. Arbab, I. Herman, and P. Spilling. An overview of Manifold and its implementation. *Concurrency: Practice and Experience*, 5(1):23–70, February 1993.
4. Backus. Can programming be liberated from the von neumann style? A functional style and its algebra of programs (1977). In *ACM Turing Award Lectures: The First Twenty Years, ACM Press Anthology Series, ACM Press, New York*. Addison-Wesley, 1987.
5. Henri E. Bal and Matthew Haines. Approaches for integrating task and data parallelism. *IEEE Concurrency*, 6(3):74–84, July/September 1998.
6. S. Bromling, S. MacDonald, J. Anvik, J. Schaeffer, D. Szafron, and K. Tan. Pattern-based parallel programming, August 2002. 2002 International Conference on Parallel Programming (ICPP-02), Vancouver, British Columbia, August 2002.
7. S. Campa and M. Danelutto. A framework for orthogonal data and control parallelism exploitation. In *Proceedings of ICCSA 2004*, Springer Verlag, LNCS, Vol. 3046, pages 1295–1300, August 2004.
8. Murray Cole. *Algorithmic Skeletons: structured management of parallel computation*. Monograms. Pitman/MIT Press, Cambridge, MA, 1989.

9. Manuel Díaz, Bartolomé Rubio, Enrique Soler, and José M. Troya. Integrating task and data parallelism by means of coordination patterns. *Lecture Notes in Computer Science*, 2026:16, 2001.
10. Ian Foster, David R. Kohr, Jr., Rakesh Krishnaiyer, and Alok Choudhary. A library-based approach to task parallelism in a data-parallel language. *Journal of Parallel and Distributed Computing*, 45(2):148–158, 15 September 1997.
11. H. Kuchen. A skeleton library. *Lecture Notes in Computer Science*, 2400:620–628, 2002.
12. H. Kuchen and M. Cole. The integration of task and data parallel skeletons. *Parallel Processing Letters*, 12(2):141, June 2002.
13. G. A. Papadopoulos and F. Arbab. Control-driven coordination programming in shared dataspace. *Lecture Notes in Computer Science*, 1277:247, 1997.
14. L. Rauchwerger, F. Arzu, and K. Ouchi. Standard templates adaptive parallel library (STAPL). *Lecture Notes in Computer Science*, 1511, 1998.

Empirical Parallel Performance Prediction from Semantics-Based Profiling

Norman Scaife[1], Greg Michaelson[2], and Susumu Horiguchi[3]

[1] VERIMAG, Centre Equation, 2, Ave de Vignat, 38610 Giers, France
Norman.Scaife@imag.fr
[2] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, Scotland, EH14 4AS
greg@cee.hw.ac.uk
[3] Department of Computer Science, Graduate School of Information Sciences, Tohoku University, Aobayama 6-3-09, Sendai 980-8579, Japan
susumu@ecei.tohoku.ac.jp

Abstract. The PMLS parallelizing compiler for Standard ML is based upon the automatic instantiation of algorithmic skeletons at sites of higher order function use. PMLS seeks to optimise run-time parallel behaviour by combining skeleton cost models with Structural Operational Semantics rule counts for HOF argument functions. In this paper, the formulation of a general rule count cost model as a set of over-determined linear equations is discussed, and their solution by singular value decomposition, and by a genetic algorithm, are presented.

1 Introduction

The PMLS (Parallelising ML with Skeletons) compiler for Standard ML [9] translates instances of a small set of common higher-order functions (HOFs) into parallel implementations of *algorithmic skeletons*. As part of the design of the compiler, we wish to implement performance-improving transformations guided by dynamic profiling. We contend that the rules that form the dynamic semantics of Standard ML provide an ideal set of counting points for dynamic profiling since they capture the *essence* of the computation at an appropriate level of detail. They also arise naturally during the evaluation of an SML program, eliminating difficult decisions about where to place counting points. Finally, the semantics provides an implementation independent basis for counting.

Our approach follows work by Bratvold [4] who used SOS rule counting, plus a number of other costs, to obtain sequential performance predictions for unnested HOFs. Bratvold's work built on Busvine's sequential SML to Occam translator for linear recursion [5] and was able to relate abstract costs in the SML prototype to specific physical costs in the Occam implementation.

Contemporaneous with PMLS, the FAN framework[2] uses costs to optimise skeleton use through transformation. FAN has been implemented within META[1] and applied to Skel-BSP, using BSP cost models and parameterisations. However, costs of argument functions are not derived automatically.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 781–789, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Alt *et al.*[3] have explored the allocation of resources to Java skeletons in computational Grids. Their skeleton cost models are instantiated by counting instruction executions in argument function byte code and applying an instruction timing model for the target architecture. As in PMLS, they solve linear equations of instruction counts from sequential test program runs to establish the timing model. However, the approach does not seem to have been realised within a compiler.

Hammond *et al.*[7] have used Template Haskell to automatically select skeleton implementations using static cost models at compile time. This approach requires substantial programmer involvement, and argument function costs are not derived automatically.

The main goal of our work is to provide predictions of sequential SML execution times to drive a transformation system for an automated parallelizing compiler. In principle, purely static methods may be used to derive accurate predictions, but for very restricted classes of program. From the start, we wished to parallelise arbitrary SML programs and necessarily accepted the limitations of dynamic instrumentation, in particular incomplete coverage and bias in test cases leading to instability and inaccuracy in predictions. However, we do not require predictions to be highly accurate so long they order transformation choices correctly.

In the following sections, we present our method for statistical prediction of SML based on the formal language definition, along with a set of test programs. We discuss the accuracy of our method and illustrate its potential use through a simple example program.

2 Semantic Rules and Performance Prediction

The SML definition[10] is based on Structural Operational Semantics (SOS) where the evaluation of a language construct is defined in terms of the evaluation of its constituent constructs. Our methodology for dynamic profiling is to set up a dependency between rule counts and program execution times, and solve this system on a learning-set of programs designated as "typical".

Suppose there are N rules in an SOS and we have a set of M programs. Suppose that the time for the ith program on a target architecture is T_i, and that the count for the jth rule when the ith program is run on a sequential SOS-based interpreter is R_{ij}. Then we wish to find weights W_j to solve the M equations:

$$R_{i1}W_1 + R_{i2}W_2 ... + R_{iN}W_N = T_i$$

This linear algebraic system can be expressed in matrix form as:

$$RW = T \tag{1}$$

Then given a set of rule counts for a new program P we can calculate a good prediction of the time on the target architecture T_P from:

$$R_{P1}W_1 + R_{P2}W_2 ... + R_{PN}W_N = T_P \tag{2}$$

These are then substituted into skeleton cost models. For the currently supported list HOFs `map` and `fold`, the models take the very simple form:

$$par\ cost\ =\ C_1 * list\ size + C_2 * send\ size + C_3 * receive\ size + C_4 * arg\ \ cost \quad (3)$$

The coefficients $C_1 ... C_4$ are determined by measurements on the target architecture, over a restricted range of a set of likely parameters[12]. We then deploy a similar fitting method to this data, relating values such as communications sizes and instance function execution times to measured run-times.

3 Solving and Predicting

We have tried to generate a set of test programs which, when profiled, include all of the rules in the operational semantics which are fired when our application is executed. We have also tried to ensure that these rules are as closely balanced as possible so as not to bias the fit towards more frequently-used rules.

We have divided our programs into a *learning* and a *test* set. The learning set consists of 99 "known" programs which cover a fair proportion of the SML language. These include functions such as mergesort, maximum segment sum, regular expression processing, random number generation, program transformation, ellipse fitting and singular value decomposition.

The test set consists of 14 "unknown" programs which, in turn, represent a fair cross-section of the learning set in terms of the sets of rules fired and the range of execution times. These include polynomial root finding, least-squares fitting, function minimisation and geometric computations. The test set was generated by classifying the entire set of programs according to type (e.g. integer-intensive computation, high-degree of recursion) and execution time. A test program was then selected randomly from each class.

To generate the design matrix R, we take the rule counts R_i^{td} and execution time T_i^{td} for top level declaration number td. The first timing T_i^0 in each repeat sequence is always ignored reducing the effect of cache-filling. The execution times $T_i^{t_i}$ are always in order of increasing number of repeats such that $T_i^x < T_i^y$ for $x < y$. Using this and knowing that outliers are always greater than normal data we remove non-monotonically increasing times within a single execution. Thus if $T_i^{td-1} < T_i^{td} < T_i^{td+1}$ then the row containing T_i^{td} is retained in the design matrix. Also, to complete the design matrix, rules in R_{all} which are not in R_i^{td} are added and set to zero.

Some rules can be trivially removed from the rule set such as those for type checking and nesting of expressions with atomic expressions. These comprise all the rules in the static semantics. However, non-significant rules are also removed by totaling up the rule counts across the entire matrix. Thus for rule r_x and a threshold θ, if:

$$\sum_{i=0}^{n} \sum_{j=0}^{t_i} R_i^j[r_x].c < \theta \sum_{i=0}^{n} \sum_{j=0}^{t_i} R_i^j[r_{max}].c \quad (4)$$

r_{max} is the most frequent rule and $R_i^j[r_k].c$ means the count for rule r_k in the list of rule counts R_i^j. Thus rules with total counts less than a threshold value times the most frequently fired rule's total count have their columns deleted from the rule matrix R. This threshold is currently determined by trial and error. The execution time vector T_n is generated from the matching execution times for the surviving rows in the rule matrix.

Fitting is then performed and the compiler's internal weights updated to include the new weights. Performance prediction is then a simple application of Equation 1, where R is the set of rules remaining after data-workup and W is the set of weights determined by fitting. For verification, the new weights are applied to the original rule counts giving reconstructed times T_{recon} and are compared with the original execution times T_n.

Once the design matrix is established using the learning set, and validated using the test set, we can then perform fitting and generate a set of weights. We have experimented with *singular value decomposition* (SVD) to solve the system as a linear least-squares problem[11]. We have also adapted one of the example programs for our compiler, a parallel *genetic algorithm* (GA) [8], to estimate the parameters for the system.

4 Accuracy of Fitting

Our compilation scheme involves translating the Standard ML core language as provided by the ML Kit Version 1 into Objective Caml, which is then compiled (incorporating our runtime C code) to target the parallel architecture. We have modified the ML Kit, which is based closely on the SML SOS, to gather rule counts directly from the sequential execution of programs.

Using an IBM RS/6000 SP2, we ran the 99 program fragments from the learning set using a modest number of repeats (from 10 to about 80, depending upon the individual execution time). After data cleanup, the resulting design matrix covered 41 apply functions[1] and 36 rules from the definition, and contained 467 individual execution times. Applying the derived weights to the original fit data gives the levels of accuracy over the 467 measured times shown in Figure 1. This table presents a comparison of the minimum, maximum, mean and standard deviation of the measured and reconstructed times for both fitting methods. The same summary is applied to the percentage error between the measured and reconstructed times.

First of all, the errors computed for both the learning and test sets look very large. However, an average error of 25.5% for SVD on the learning set is quite good considering we are estimating runtimes which span a scale factor of about 10^4. Furthermore, we are only looking for a *rough* approximation to the absolute values. When we apply these predictions in our compiler it is often the *relative* values which are more important and these are much more accurate although more difficult to quantify.

[1] Apply functions are external primitive functions called by the SML core language.

	Fit	χ^2	Time (s)	Min	Max	Mean	Std. Dev.
Learning Set	x		Measured	5.11×10^{-6}	0.00235	0.000242	0.000425
	SVD	4.1×10^{-7}	Reconstructed	-2.65×10^{-6}	0.00239	0.000242	0.000424
		% Error	0.00571%	267.0%	25.5%	41.3%	
	GA	4.9×10^{-5}	Reconstructed	5.98×10^{-8}	0.00163	0.000179	0.000247
		% Error	0.00977%	1580.0%	143.0%	249.0%	
Test Set	x		Measured	8.61×10^{-6}	0.0399	0.00221	0.0076
	SVD		Reconstructed	-8.06×10^{-5}	0.0344	0.00195	0.00656
		% Error	0.756%	836.0%	158.0%	208.0%	
	GA		Reconstructed	1.67×10^{-7}	0.01600	0.000965	0.000304
		% Error	1.56%	284.0%	67.9%	71.1%	

Fig. 1. Summary of fit and prediction accuracy

The SVD is a much more accurate fit than GA as indicated by the χ^2 value for the fit. However, the SVD fit is much less stable than the GA fit as evidenced by the presence of negative reconstructed times for SVD. This occurs at the very smallest estimates of runtime near the boundaries of the ranges for which our computed weights are accurate. The instability rapidly increases as the data moves out of this region.

5 Performance Prediction Example

As part of the PMLS project we have used *proof-planning* to construct a synthesiser which extracts HOFs from arbitrary recursive functions[6]. For example, given the following program which squares the elements of a list of lists of integers:

```
fun squares [] = []
  | squares ((h:int)::t) = h*h::squares t

fun squs2d [] = []
  | squs2d (h::t) = squares h::squs2d t
```

the synthesizer generates the six programs shown in Figure 2. Note that there is no parallelism in this program suitable for our compiler and we would expect our predictions to validate this.

We require the execution times for the instance functions to the `map` and `foldr` HOFs. We have not yet automated the collection of this data or linked the output from the performance prediction into the compiler so we present a hand analysis of this code.

Figure 3 shows the predicted instance function execution times for the two fitting methods alongside the actual measured times. The input data is a 5×5 list of lists of integers. The predictions are in roughly the correct range but differ significantly from the measured times. Despite the greater accuracy of the

```
1. val squs2d = fn x => map (fn y => map (fn (z:int) => z*z) y) x
2. val squs2d =
       fn x => foldr (fn y => fn z => (map (fn (u:int) => u*u) y::z)) [] x
3. val squs2d =
       fn x => map (fn y => foldr (fn (z:int) => fn u => z*z::u) [] y) x
4. val squs2d =
        fn x => foldr (fn y => fn z =>
                        foldr (fn (u:int) => fn v => u*u::v) [] y::z) [] x
5. val squs2d = fn x => map (fn y => squares y) x
6. val squs2d = fn x => foldr (fn y => fn z => squares y::z) [] x
```

Fig. 2. Synthesizer output for `squs2d`

V	Position	HOF	Rules	T_{SVD}	T_{GA}	$T_{measured}$
1	outer	map	21	2.63	5.56	8.61
	inner	map	8	0.79	1.40	3.36
2	outer	fold	21	4.97	6.01	9.17
	inner	map	8	0.79	1.40	3.14
3	outer	map	20	1.73	7.53	12.6
	inner	fold	15	12.5	3.66	3.71
4	outer	fold	20	4.06	7.98	11.1
	inner	fold	15	12.5	3.66	3.53
5	single	map	19	3.58	3.45	6.65
6	single	fold	19	5.91	3.90	7.97

Fig. 3. Predicted and measured instance function times (μS)

SVD fit to the learning-set data, the GA-generated weights give more consistent results compared to actual measured values. This is due to the numerical instability of the SVD fit. However, these discrepancies are sufficient to invert the execution times for nested functions. For instance, for Version 3 the inner fold instance function takes longer than the outer one, even though the outer computation encompasses the inner.

Applying the skeleton performance models to the measured instance function times, plus data on communications sizes gathered from sequential executions, gives the predicted parallel run-times for 1, 2, 4 and 8 processors, shown in Figure 4.

The GA- and SVD-predicted instance function times give identical predictions for parallel run-times. This is because the parallel performance model is in a range where the run-time is dominated by communications rather than computation. However, the P_1 predictions are erroneous. These predictions represent an extrapolation of a parallel run onto a sequential one which has no overheads such as communication. This also applies to the P_2 predictions, where these overheads are not accurately apportioned. Furthermore, the absolute values of

V	Position	HOF	P/M	P_1	P_2	P_4	P_8
1	outer	map	P	1.6000	3.230	6.480	12.990
			M	0.1423	6.806	5.279	4.910
	inner	map	P	3.2700	4.900	8.150	14.660
			M	0.2846	35.200	15.620	14.440
2	outer	fold	P	7.3700	10.940	18.070	32.340
			M	0.1617	4.204	3.101	3.634
	inner	map	P	3.2700	4.900	8.150	14.660
			M	0.3040	35.360	14.900	14.940
3	outer	map	P	1.6000	3.230	6.480	12.990
			M	0.2205	7.314	3.923	4.739
	inner	fold	P	14.2000	17.760	24.900	39.170
			M	0.3875	26.020	14.570	15.770
4	outer	fold	P	7.3700	10.940	18.070	32.340
			M	0.2344	5.058	2.907	4.047
	inner	fold	P	14.2000	17.760	24.900	39.170
			M	0.3907	23.080	13.200	16.110
5	single	map	P	1.6000	3.230	6.480	12.990
			M	0.1375	6.590	4.092	4.570
6	single	fold	P	7.3700	10.940	18.070	32.340
			M	0.1587	4.024	3.002	3.750

Fig. 4. Predicted (P) and measured (M) parallel run-times (mS)

the predictions are unreliable. For the P_8 values, some are accurate but some are out by an order of magnitude. The most relevant data in this table is the ratio between the P_4 and P_8 values. This, in most cases, increases as the number of processors increases, indicating slowdown.

6 Conclusions

Overall, our experimentation gives us confidence that combining automatic profiling with cost modeling is a promising approach to performance prediction. We now intend to use the system as it stands in implementing a performance-improving transformation system for a subset of the SML language. As well as exploring the automation of load balancing, this gives us a further practical way to assess the broader utility of our approach.

While we have demonstrated the feasibility of semantics-based profiling for an entire extant language, further research is needed to enable more accurate and consistent predictions of performance from profiles. Our work suggests a number of areas for further study.

It would be useful to identify which semantic rules counts are most significant for predicting run times, through standard statistical techniques for correlation and factor analyses. Focusing on significant rules would reduce profiling

overheads and might enable greater stability in the linear equation solutions. Furthermore, non-linear costs might be introduced into the system, relating profile information and runtime measurements. The system would no longer be in matrix form and would require the use of generalised function minimisation instead of deterministic fitting. Predictions might also be made more accurate by investigating the effects of optimisations employed in the back end compiler, which fundamentally affect the nature of the association between the language semantics and implementation. Our studies to date have been of very simple functions and of unrelated substantial exemplars: it would be worth systematically exploring the relationship between profiles and run-times for one or more constrained classes of recursive constructs, in the presence of both regular and irregular computation patterns. Finally, aspects of implementation which are subsumed in the semantics notation might be modeled explicitly, in particular the creation and manipulation of name/value associations which are hidden behind the semantic notion of environment.

Acknowledgments

This work was supported by Postdoctoral Fellowship P00778 of the Japan Society for the Promotion of Science (JSPS) and by UK EPSRC grant GR/L42889.

References

1. M. Aldinucci. Automatic Program Transformation: The META Tool for Skeleton-based Languages. In S. Gorlatch and C. Lengauer, editors, *Constructive Methods for Parallel Programming*, volume 10 of *Advances in Computation: Theory and Practice*. NOVA Science, 2002.
2. M. Aldinucci, S. Gorlatch, C. Lengauer, and S. Pelegatti. Towards Parallel Programming by Transformation: The FAN Skeleton Framework. *Parallel Algorithms and Applications*, 16(2-3):87–122, March 2001.
3. M. Alt, H. Bischof, and S. Gorlatch. Program Development for Computational Grids Using Skeletons and Performance Prediction. *Parallel Processing Letters*, 12(2):157–174, 2002.
4. T. Bratvold. *Skeleton-based Parallelisation of Functional Programmes*. PhD thesis, Dept. of Computing and Electrical Engineering, Heriot-Watt University, 1994.
5. David Busvine. Implementing Recursive Functions as Processor Farms. *Parallel Computing*, 19:1141–1153, 1993.
6. A. Cook, A. Ireland, G. Michaelson, and N. Scaife. Deriving Applications of Higher-Order Functions through Proof Planning. *Formal Aspects of Computing*, accepted Nov. 2004.
7. K. Hammond, J. Berthold, and R. Loogen. Automatic Skeletons in Template Haskell. *Parallel Processing Letters*, 13(3):413–424, 2003.
8. G. Michaelson and N.Scaife. Parallel functional island model genetic algorithms through nested skeletons. In M. Mohnen and P. Koopman, editors, *Proceedings of 12th International Workshop on the Implementation of Functional Languages*, pages 307–313, Aachen, September 2000.

9. G. Michaelson and N. Scaife. Skeleton Realisations from Functional Prototypes. In F. Rabhi and S. Gorlatch, editors, *Patterns and Skeletons for Parallel and Distributed Computing*. Springer, 2003.
10. R. Milner, M. Tofte, and R. Harper. *The Definition of Standard ML*. MIT, 1990.
11. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. CUP, 2nd edition, 1992.
12. N. R. Scaife. *A Dual Source, Parallel Architecture for Computer Vision*. PhD thesis, Dept. of Computing and Electrical Engineering, Heriot-Watt University, 1996.

Dynamic Memory Management in the *Loci* Framework

Yang Zhang and Edward A. Luke

Department of Computer Science and Engineering,
Mississippi State University, Mississippi State, MS 39762, USA
{fz15, luke}@cse.msstate.edu

Abstract. Resource management is a critical concern in high-performance computing software. While management of processing resources to increase performance is the most critical, efficient management of memory resources plays an important role in solving large problems. This paper presents a dynamic memory management scheme for a declarative high-performance data-parallel programming system — the *Loci* framework. In such systems, some sort of automatic resource management is a requirement. We present an automatic memory management scheme that provides good compromise between memory utilization and speed. In addition to basic memory management, we also develop methods that take advantages of the cache memory subsystem and explore balances between memory utilization and parallel communication costs.

1 Introduction

In this paper we discuss the design and implementation of a dynamic memory management strategy for the declarative programming framework, *Loci* [1, 2]. The *Loci* framework provides a rule-based programming model for numerical and scientific simulation similar to the *Datalog* [3] logic programming model for relational databases. In *Loci*, the arrays typically found in scientific applications are treated as relations, and computations are treated as transformation rules. The framework provides a planner, similar to the FFTW [4] library, that generates a schedule of subroutine calls that will obtain a particular user specified goal. *Loci* provides a range of automatic resource management facilities such as automatic parallel scheduling for distributed memory architectures and automatic load balancing. The *Loci* framework has demonstrated predictable performance behavior and efficient utilization of large scale distributed memory architectures on problems of significant complexity with multiple disciplines involved [2]. *Loci* and its applications are in active and routine use by engineers at various NASA centers in the support of rocket system design and testing.

The *Loci* planner is divided into several major stages. The first stage is a dependency analysis which generates a dependency graph that describes a partial ordering of computations from the initial facts to the requested goal. In the second stage, the dependency graph is sub-divided into functional groups that are further partitioned into a collection of directed acyclic graphs (DAGs). In the third stage, the partitioned graphs

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 790–797, 2005.
© Springer-Verlag Berlin Heidelberg 2005

are decorated with resource management constraints (such as memory management constraints). In the forth stage a proto-plan is formed by determining an ordering of DAG vertices to form computation super-steps. (In the final parallel schedule, these steps are similar to the super-steps of the Bulk Synchronous Parallel (BSP) model [5].) The proto-plan is used to perform analysis on the generation of relations by rules as well as the communication schedule to be performed at the end of each computation step in the fifth and sixth stages (existential analysis and pruning), as described in more detail in this recent article [2]. Finally the information collected in these stages is used to generate an execution plan in the seventh stage. Dynamic memory management is primarily implemented as modifications to the third and fourth stages of *Loci* planning.

2 Related Work

The memory system and its management has been studied extensively in the past. These studies are on various different levels. When designing the memory management subsystem for *Loci*, we are mostly interested in designing a memory management strategy and not in low level allocator designs. The programming model in *Loci* is declarative, which means the user does not have direct control of allocation. Also one major goal of the *Loci* framework is to hide irrelevant details from the user. Therefore we are interested in designing an automatic memory management scheme. Garbage collection [6] is the most prevalent automatic memory management technique. Useless memory blocks are treated as garbage and are recycled periodically by the run-time system. A non-traditional method for managing memory resources in the context of scheduling operators in continuous data streams [7] shows how scheduling order can effect overall memory requirements. They suggest an optimal strategy in the context of stream models. Region inference [8] is a relatively new form of automatic memory management. It relies on static program analysis and is a compile-time method and uses the region concept. The compiler analyzes the source program and infers the allocation. In addition to being fully automatic, it also has the advantage of reducing the run-time overhead found in garbage collection. Garbage collection typically works better for small allocations in a dynamic environment. While in *Loci*, the data-structures are often static and allocations are typically large. Thus, the applicability of garbage collection to this domain is uncertain. Instead of applying traditional garbage collection techniques, we have adopted a strategy that shares some similarities to the region inference techniques as will be described in the following sections.

3 Basic Dynamic Memory Management

In *Loci*, relations are stored in value containers. These containers are the major source of memory consumption. Therefore the management of allocation and deallocation of these containers is the major focus of our memory management scheme. A simple way to manage the lifetime of these containers is preallocation. In this approach we take advantage of the *Loci* planner's ability to predict the sizes of the containers in advance. In the preallocation scheme, all containers are allocated at the beginning and recycled only at the end of the schedule. While this scheme is simple and has little

run-time overhead, it does not offer any benefits for saving space. Scientific applications for which *Loci* is targeted tend to have large memory requirements. The primary goal of the management is therefore to reduce the peak memory requirement so that larger problems can be solved on the same system. Preallocation obviously fails this purpose.

Since *Loci* planner generates an execution schedule from the partitioned dependency graph (the multi-level graph), a simple approach to incorporating appropriate memory scheduling would be to incorporate relevant memory management operations into this graph. Then, when the graph is compiled, proper memory management instructions are included into the schedule that will be invoked in execution. We refer this process of including memory management instructions into the dependency graph as graph decoration. Thus memory management for *Loci* becomes the graph decoration problem. The multi-level graph for a real application is likely to be complex. For example, multiple nested iterations and conditional specifications, recursions, etc. could also be involved. A global analysis of the graph is performed to determine the lifetime of all containers in the schedule [9].

4 Chomping

Chomping is a technique we used in *Loci* to optimize the cache performance. The idea of chomping is borrowed from the commonly known loop scheduling technique: strip mining. In *Loci*, relations, the primary data abstractions, are collections of attributes that are stored in array-like containers that represent aggregations of values. Since these containers dominate the space consumed by Loci applications, they are ideal candidates for memory savings by data partitioning. Consider the rule chain in Fig. 1. Relation `A` is the source to the chain and `D` is the final derived relation; `B` and `C` are intermediate relations. We can break the rules in the chain into small sub-computations. In each of these sub-computation, only part of the derived relations are produced. This implies for any intermediate relations, only partial allocation of their container is required. Because these partial allocations can be made small, they enhance cache utilization and can further reduce the memory requirement. However, because of the existence of non-affine memory references, we cannot group arbitrary rules into rule chains that can be chomped. In *Loci*, we use a heuristic search to identify suitable chains in the multi-level graph and apply chomping only to them [9]. Breaking computations into smaller intermediate segments not only reduces absolute memory allocation requirements, but also helps to reduce fragmentation by reusing a pool of small uniformly sized memory segments.

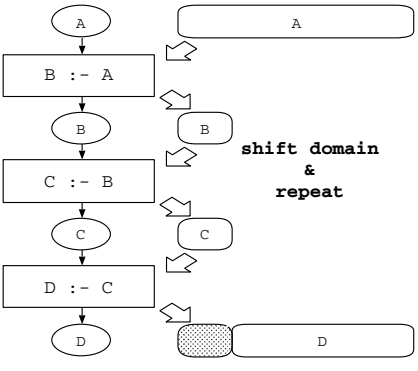


Fig. 1. The Chomping Idea

5 Memory Utilization and Parallel Communication Costs

In section 3, we transformed the memory management into a graph decoration problem. However the graph decoration only specifies a dependencies between memory management and computation. It is up to the *Loci* planner to generate a particular execution order that satisfies this dependence relationship. From the memory management point of view, the order to schedule allocation and deallocation affects the peak memory requirement of the application. On the other hand, *Loci* planner can produce a data-parallel schedule. In the data-parallel model, after each super-step, processors need to synchronize data among the processes. From the communication point of view, different schedules may create different numbers of synchronization points. While the number of synchronization points does not change the total volume of data communicated, increased synchronization does reduce the opportunity to combine communication schedules to reduce start-up costs and latency. Thus with respect to parallel overhead, less synchronization is preferred.

Figure 2 shows the effect of different scheduling of a DAG. Schedule one is greedy on computation, a rule is scheduled as early as possible. Therefore schedule one has less synchronization points. Schedule two is greedy on memory, a rule is scheduled as late as possible. Therefore derived relations are spread over more super-steps, hence more synchronization points are needed.

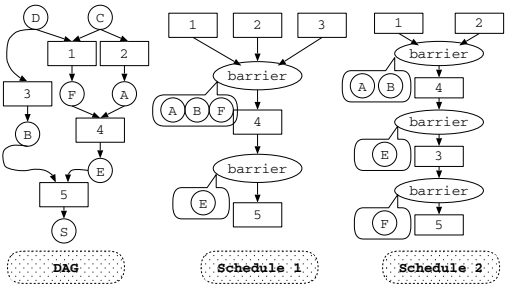


Fig. 2. Different Scheduling for a DAG

A trade-off therefore exists in the *Loci* planner. In order to optimize memory utilization and reduce the peak memory requirement, the planner will typically generate a schedule with more synchronization points, and therefore increase the communication start-up costs and slow down the execution. Attempting to minimize the synchronization points in a schedule results in a fast execution, but with more memory usage. Such trade-off can be customized under different circumstances. For example, if memory is the limiting factor, then a memory optimization schedule is preferred. In this case, speed is sacrificed for getting the program run within limited resources. On the other hand, if time is the major issue, then a computation greedy schedule is preferred, but users have to supply more memory to obtain speed. In the *Loci* planner, we have implemented two different scheduling algorithms. One is a simple computation greedy scheduling algorithm, which minimizes the total synchronization points. The other one is a memory greedy scheduling algorithm. It relies on heuristics to attempt to minimize the memory usage. Users of *Loci* can instruct the planner to choose either of the two policies.

The scheduling infrastructure in the *Loci* planner is priority based. *Loci* planner schedules a DAG according to the weight of each vertex. In this sense, scheduling policies can be implemented by providing different weights to the vertices. We provide a heuristic for assigning vertices weight that attempts to minimize the memory utilization for the schedule. The central idea of the heuristic is to keep a low memory usage in

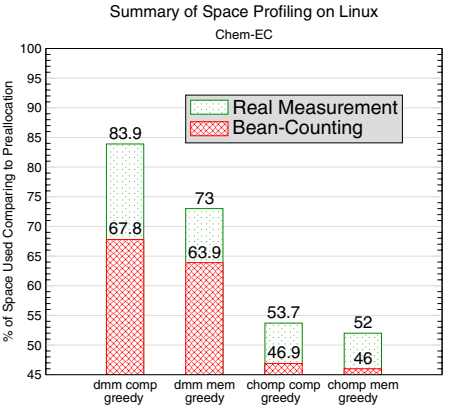
each scheduling step. Given a DAG with memory management decoration, rules that do not cause memory allocation have the highest priority and are scheduled first. They are packed into a single step in the schedule. If no such rules can be scheduled, then we must schedule rules that cause allocation. The remaining rules are categorized. For any rule that causes allocation, it is possible that it also causes memory deallocation. We schedule one such rule that causes most deallocations. If multiple rules have the same number of deallocations, we schedule one that causes fewest allocations. Finally, we schedule all rules that do not meet the previous tests, one at a time with the fewest outgoing edges from all relations that it produces. This is based on the assumption that the more outgoing edges a relation has in a DAG, the more places will it be consumed, hence the relation will have a longer lifetime. We used a sorting based algorithm [9] in Loci for computing vertex priority based on the heuristics described above for memory minimization.

6 Experimental Results

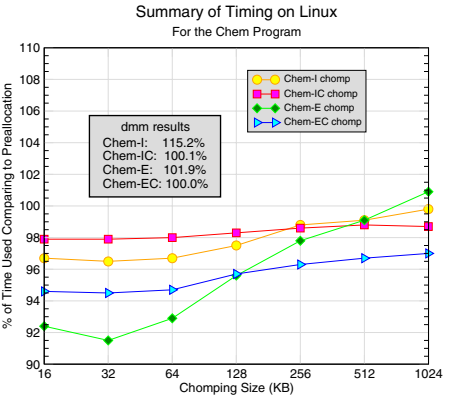
In this section, we present some of our measurements for the dynamic memory management in *Loci*. The CHEM program is used as a benchmark. CHEM is a finite-rate non-equilibrium Navier-Stokes solver for generalized grids fully implemented using the *Loci* framework. CHEM can be configured to run in several different modes, they are abbreviated as Chem-I, Chem-IC, Chem-E, and Chem-EC in the following figures and tables. An IBM Linux Cluster (total 1038 1GHz and 1.266GHz Pentium III processors on 519 nodes, 607.5 Gigabytes of RAM) is used in the measurement. In addition to take the measurement of the real memory usage, we also record the bean-counting memory usage numbers. (By bean-counting we mean tabulating the exact amount of memory requested from the allocator. It is shown as a reference as we use GNU GCC's allocator in *Loci*.) In the measurement, we are comparing the results with the preallocation scheme mentioned in section 3, as the preallocation scheme represents the upper-bound for space requirement and the lower-bound for run-time management overhead.

We did extensive profiling of the memory utilization on various architectures. Figure 3(a) shows a measurement of Chem-EC on a single node on the cluster. The "dmm" in the figure means the measurement was performed with the dynamic memory management enabled; "chomp" means chomping was also activated in the measurement in addition to basic memory management. As can be found from Fig. 3(a), when combining with memory greedy scheduling and chomping, the peak memory usage is reduced to at most 52% of preallocation peak memory usage. The actual peak memory depends also on the design of the application. We noticed that for some configurations, the difference between the real measurement and the bean-counting is quite large. We suspect that this is due to the quality of the memory allocator. We also found that under most cases, using chomping and memory greedy scheduling will help to improve the memory fragmentation problem. Because in these cases, the allocations are possibly much smaller and regular.

Figure 3(b) shows one timing result for chomping on a single node on the cluster. The result shows different chomping size for different CHEM configurations. Typically using chomping increases the performance, although no more than 10% in our case.



(a) Space Profiling on Linux



(b) Timing on Linux

Fig. 3. Space and Timing Measurement

The benefit of chomping also depends on the *Loci* program design, the more computations are chomped, the more benefit we will have. The box in Fig. 3(b) shows the speed of dynamic memory management alone when compared to the preallocation scheme. This indicates the amount of run-time overhead incurred by the dynamic memory management. Typically they are negligible. The reason for the somewhat large overhead of Chem-I under "dmm" is unknown at present and it is possible due to random system interactions.

To study the effect of chomping under conditions where the latencies in the memory hierarchy are extreme, we performed another measurement of chomping when virtual memory is involved. We run CHEM on a large problem such that the program had significant access to disk through virtual memory. We found in this case, chomping has superior benefit. Schedule with chomping is about 4 times faster than the preallocation schedule or the schedule with memory management alone. However the use of virtual memory tends to destroy the performance predictability and thus it is desirable to avoid virtual memory when possible. For example, a large memory requirement can be satisfied by using more processors. Nevertheless, this experiment showed an interesting feature of chomping. Chomping may be helpful when we are constrained by system resources.

Finally we present one result of the comparison of different scheduling policies in table 1. The measurement was performed on 32 processors of our parallel cluster. We

Table 1. Mem vs. Comm under dmm on Linux Cluster

	memory usage (MB)		sync		time
	real	bean-counting	points	time (s)	ratio(%)
comp greedy	372.352	174.464	32	3177.98	1
mem greedy	329.305	158.781	50	3179.24	1.0004

noticed the difference of peak memory usage between computation greedy and memory greedy schedule is somewhat significant, however the timing results are almost identical albeit the large difference in the number of synchronization points. We attribute this to the fact that CHEM is computationally intensive, the additional communication start-up costs do not contribute significantly to the total execution time. This suggests for computational intensive application, the memory greedy scheduling is a good overall choice, as the additional memory savings do not incur undue performance penalty. For more communication oriented applications, the difference of using the two scheduling policies may be more obvious. In another measurement, we artificially ran a small problem on many processors such that parallel communication is a major overhead. We found the synchronization points in the memory greedy schedule is about 1.6 times more than the one in computation greedy schedule and the execution time of memory greedy schedule increased roughly about 1.5 times. Although this is an exaggerated case, it provided some evidence that such trade-off does exist. However, for scaling small problems, memory resources should not be a concern and in this case the computation greedy schedule is recommended.

7 Conclusions

The study presented in this paper provides a dynamic memory management infrastructure for the *Loci* framework. We transformed memory management to a graph decoration problem. The proposed approach utilized techniques to improve both cache utilization and memory bounds. In addition, we studied the impact of memory scheduling on parallel communication overhead. Results show that the memory management is effective and is seamlessly integrated into the *Loci* framework. Combining the memory management with chomping, the resulting schedule is typically faster and space efficient. The aggregation performed by *Loci* also facilitates the memory management and cache optimization. We were able to use *Loci*'s facility of aggregating entities of like type as a form of region inference. The memory management is thus simplified as managing the lifetime of these containers amounted to managing the lifetimes of aggregations of values. In this sense, although *Loci* supports fine-grain specification [2], the memory management does not have to be at the fine-grain level. This has some similarity with the region management concept. The graph decoration resembles the static program analysis performed by the region inference memory management, although much simpler and is performed at run-time. The scheduling policies implemented in *Loci* are currently specified by users. As a future work, it is possible to extend this and make *Loci* aware of the scheduling policies itself. We imagine there are several different ways to achieve this. In *Loci*, we can estimate the overall computation and communication time and the memory consumption before the execution plan is run. Therefore we can infer an appropriate scheduling policy in *Loci* and thus does not require the user being aware of this choice. A more sophisticated way would be to generate two schedules (one for memory minimization and the other for communication overhead minimization) and switch between them at runtime. Since it is possible that some containers would be dynamically resized at runtime, the estimation at the scheduling phase could be imprecise. If we have two schedules, we can dynamically measure the cost at runtime and switch

to an appropriate schedule when necessary. This scheme requires some amount of coordinations between different schedules and is much harder than the previous scheme. But as we observed, current *Loci* applications are typically computation bounded and therefore this feature is less critical.

Acknowledgments

The financial support of the National Science Foundation (ACS-0085969), NASA GRC (NCC3-994), and NASA MSFC (NAG8-1930) is gratefully acknowledged. In addition we would like to thank the anonymous reviewers for their excellent suggestions.

References

1. Luke, E.A.: Loci: A deductive framework for graph-based algorithms. In Matsuoka, S., Oldehoeft, R., Tholburn, M., eds.: Third International Symposium on Computing in Object-Oriented Parallel Environments. Number 1732 in Lecture Notes in Computer Science, Springer-Verlag (1999) 142–153
2. Luke, E.A., George, T.: Loci: A rule-based framework for parallel multi-disciplinary simulation synthesis. Journal of Functional Programming, Special Issue on Functional Approaches to High-Performance Parallel Programming (to appear) available at: `http://www.erc.msstate.edu/~lush/publications/LociJFP2005.pdf`.
3. Ullman, J.: Principles of Database and Knowledgebase Systems. Computer Science Press (1988)
4. Frigo, M., Johnson, S.G.: FFTW: An adaptive software architecture for the FFT. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. Volume 3., Seattle, WA (1998) 1381–1384
5. Valiant, L.G.: A bridging model for parallel computation. Communications of the Association for Computing Machinery **33** (1990) 103–111
6. Wilson, P.R.: Uniprocessor garbage collection techniques. In: Proceedings of International Workshop on Memory Management, St. Malo, France, Springer-Verlag (1992)
7. Babcock, B., Babu, S., Datar, M., Motwani, R.: Chain: Operator scheduling for memory minimization in data stream systems. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD 2003), San Diego, California (2003)
8. Tofte, M., Birkedal, L.: A region inference algorithm. Transactions on Programming Languages and Systems (TOPLAS) **20** (1998) 734–767
9. Zhang, Y.: Dynamic memory management for the Loci framework. Master's thesis, Mississippi State University, Mississippi State, Mississippi (2004)

On Adaptive Mesh Refinement for Atmospheric Pollution Models

Emil M. Constantinescu and Adrian Sandu

Department of Computer Science, Virginia Polytechnic Institute
and State University, Blacksburg, VA 24061
{emconsta, asandu}@cs.vt.edu

Abstract. This paper discusses an implementation of an adaptive resolution system for modeling regional air pollution based on the chemical transport model STEM. The grid adaptivity is implemented using the generic tool Paramesh. The computational algorithm uses a decomposition of the domain, with the solution in different sub-domains computed at different spatial resolutions. We analyze the parallel computational performance versus the accuracy of long time simulations.

Keywords: Air Pollution Modeling, Adaptive Mesh Refinement.

1 Introduction

Inadequate grid resolution can be an important source of errors in air pollution modeling (APM) where large spatial gradients of tracer concentrations result from the complex interactions between emissions, meteorological conditions, and nonlinear atmospheric chemistry [9].

Chock et. al. [3] studied the effects of grid resolution on model predictions of non-homogeneous atmospheric chemistry. They concluded that increasing the grid size leads to a reduction of the suppression of ozone (O_3) in the presence of high nitrogen oxides ($NO_X = NO + NO_2$), and a decrease in the strength of the NO_X inhibition effect. O_3 loses nearly all the detail near the emission source in the coarse grid case.

A popular multi-resolution approach in air pollution and meteorological modeling is static nesting of finer grids into coarser grids. This approach requires apriori knowledge of where to place the high resolution grids inside the modeling domain; but it does not adjust to dynamic changes in the solution during simulation. In many practical situations the modeler "knows" where higher resolution is needed, e.g. above industrial areas.

In this paper we investigate the parallel performance and accuracy improvements for an application of adaptive mesh refinement (AMR) for modeling regional air pollution. The grid adapts dynamically during the simulation, with the purpose of controlling the numerical spatial discretization error. Unlike uniform refinement, adaptive refinement is more economical. Unlike static grid nesting,

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 798–805, 2005.
© Springer-Verlag Berlin Heidelberg 2005

with AMR it is not necessary to specify in advance which areas need higher resolution; what is required is to define a refinement criterion, which is then used by the code to automatically adjust the grid.

The use of generic AMR tools (Paramesh [6, 7]) allows us to harness the power of parallel computers for regional air pollution simulations. Parallel computing is needed as the higher the resolution, the more expensive the simulation becomes.

The paper is organized as follows. Sec. 2 gives an overview of previous work. A brief description of the static mesh APM application and of the AMR system used in this paper is given in Sec. 3. The refinement criterion used in this paper is described in detail in Sec. 3.1. Numerical results are shown in Sec. 4, and Sec. 5 presents conclusions and future research directions.

2 Previous Work

Adaptive meshes have been used in the study of pollutant dispersion in atmosphere [10, 9, 5]. In this section we will discuss some atmospheric AMR applications.

The Ph.D. dissertation of van Loon [10] is focused on numerical methods for smog prediction. The model developed, CWIROS, has 4 vertical layers and its horizontal domain covers all Europe. The horizontal resolution is 60×60 Km, with 4 levels of refinement. The spatial error estimator uses the curvature of the concentration fields. Specifically, for each column the horizontal curvature is estimated for each species and normalized. The column is flagged for refinement if the error estimator is larger than a user-prescribed tolerance.

Srivastava, McRae and Odman [9] discuss a very interesting approach to grid adaptivity (DSAGA-PPM) for simulating reactive atmospheric pollutants. DSAGA-PPM uses horizontal (2D) adaptivity and employs a constant number of grid nodes. This keeps the total computational time for a simulation manageable. A weight function is defined by a linear combination of curvatures of different chemical species. Based on this weight function the grid is adapted.

3 Implementation Considerations

In this section we describe the numerical and software components of our application. First, we discuss the science application and next we briefly present the AMR approach.

The core science application used in this paper is the state–of–the–art regional APM, STEM [2]. The original code uses a fixed grid, with all the data structures being multidimensional arrays.

STEM solves the advection-diffusion-reaction equation for N species on a 3-D domain:

$$\frac{\partial}{\partial t}c_i + \nabla \cdot (uc_i) = \nabla \cdot (D\nabla c_i) + f(c_i), \text{ with } i = 1 \ldots N.$$

The equation is solved using an operator splitting approach. STEM uses linear finite difference discretizations of the transport terms and Rosenbrock methods for solving stiff chemistry [8].

Paramesh offers the infrastructure for AMR on a 2D structured grid.Paramesh is a parallel AMR Fortran toolkit developed by P. MacNeice and K. Olson at the NASA Goddard Space Flight Center [6, 7]. The adaptive resolution is based on a Schwarz-type domain decomposition, with a single Schwarz iteration. We use a two-dimensional (horizontal) grid refinement approach. All the data associated with a column in the original STEM (referred to as STEM variables) are assigned to a mesh-point (cell) in Paramesh, including geographical and meteorological data and species concentrations along the z axis.

The domain is divided into blocks, each containing 6×6 cells plus two guard-cells along each block boundary [1]. At the coarse level, each cell has a resolution of 80×80 Km. For the TraceP (described in Sec. 4.1) simulation over East Asia, the computational domain is covered by 15×10 blocks. At the finest level (level 4) each cell has a resolution of 10×10 Km. Data are linearly interpolated between refinement levels during each mesh refinement/derefinement operation.

During the simulation data available in STEM-specific data types need to be copied into Paramesh data types. The initial species concentrations and geographical information are provided at coarse level at the beginning of the simulation. Meteorological fields, boundary conditions and surface emissions are updated every hour. All data are given at the coarse level, except for the emission inventories which are provided at a fine resolution (10×10 Km).

Experimentally, we noticed a loss in accuracy associated with block refinement near the physical boundary due to the fact that boundary data are available at coarse level only. A refinement restriction was applied to blocks neighboring the domain boundary, such that they are maintained at coarse levels.

The regriding process is handled by Paramesh. Each time the regriding is performed, blocks on each processor are refined or derefined according to one of our criteria and then migrated to other processors if necessary with the goals of load balancing and data locality.

3.1 Refinement Criteria

The estimation of the spatial error in a $NXB \times NYB = N^2$ cells horizontal block (at vertical level k) is done based on the horizontal curvature of the concentration field c at each point (i, j, k) in space

$$\begin{aligned} err_{i,j,k} = &|c_{i+1,j,k} - 2c_{i,j,k} + c_{i-1,j,k}| + \\ &+ |c_{i,j+1,k} - 2c_{i,j,k} + c_{i,j-1,k}|, \end{aligned}$$

and by taking the root mean square value normalized by the maximum concentration inside the block

$$ERR_k(c) = \begin{cases} \frac{\sqrt{\sum_{i,j} err_{i,j,k}^2}}{N \cdot \max_{i,j} c_{i,j,k}} & \text{if } \max_{i,j} c_{i,j,k} \geq Atol \\ 0 & \text{if } \max_{i,j} c_{i,j,k} < Atol \end{cases}$$

Note that the error is ignored if the concentration inside the block is below a user-prescribed level, $Atol$. The total error estimate in a column is taken to be the maximum error among all layers $ERR(c) = \max_k ERR_k(c)$. The block is flagged for refinement if $ERR(c) \geq uptol$ and for derefinement if $ERR(c) \leq lowtol$.

The model calculates the concentrations of a large number of trace species and the refinement pattern depends on which concentrations are used for error estimation. We consider a multiple species criterion, focusing on O_3, formaldehyde ($HCHO$) and NO_X compounds – the main precursors of O_3. A weighted combination of the regarded species is considered: $w_1 NO + w_2 NO_2 + w_3 O_3 + w_4 HCHO$, with $w_{1,2} = 35\%$ and $w_{3,4} = 15\%$. The error for a mesh-point, based on ℓ species $i_1, \cdots i_\ell$ (in our case $\ell = 4$) the error is estimated by

$$ERR(c_{i_1} \dots c_{i_\ell}) = \sqrt{\frac{1}{\ell} \sum_{j=1}^{\ell} w_j ERR(c_{i_j})^2} \quad .$$

Figure 1.d shows the refined grid pattern corresponding to 0 GMT March 1^{st}, 2001 over East Asia, TraceP conditions with $uptol = 0.25$, $lowtol = 0.1$. The grid is refined in the areas of high emissions, i.e. above industrial regions in China, Japan and Korea. The refinement criteria is applied at simulated hourly increments. In our experiments, we regrid every three hours.

4 Results

In this section we analyze the performance of the parallel implementation for work load and accuracy. Sec. 4.1, describes the experimental setting and Sec. 4.2 discusses the results.

4.1 Experimental Setting

The test problem is a real-life simulation of air pollution in East Asia in support of the TraceP [4] field experiment. The TraceP (NASA TRAnsport and Chemical Evolution over the Pacific) field experiment was conducted in East Asia. The meteorological fields, boundary values and emission rates correspond to TraceP starting at 0 GMT of March 4^{th}, 2001 for one week of simulation time. Due to the fact that our initial data are not smooth enough, an imminent transient phase tends to occur if one should start refining from the coarse level. Instead, we simulated two days, starting with March 1^{st}, 2001, at the finest level and after that we applied the mesh refinement criterion and allowed the grid to coarsen. The accuracy and timing results were taken into consideration after three days of simulation (two at the finest level and one for system adjusting to a relative steady state).

The simulated region covers 7200×4800 Km. At the coarse level (each cell with 80×80 Km) there are 150 blocks, each containing 6×6 cells. At the finest level, level 4 (10×10 Km) there are 5058 working blocks (182,088 mesh points). Each cell holds a column of 18 layers with 2340 of STEM variables.

The simulations are performed on Virginia Tech's System X, the fastest academic supercomputer in the world. It has 1100 node Apple XServe G5 dual processor, 4 Gb of RAM per node. The interconnect consists of InfiniBand switches (primary) and Cisco 4506 Gigabit Ethernet (secondary). We were unable to accurately measure execution times for long simulations but we were able to make an estimation of the scalability based on short one simulated hour runs.

4.2 Numerical/Parallel Performance

The timing results for 1 simulated hour at the finest refinement level on 16, 32, 64 and 96 processors are presented in table 1.a. Considering the fact that the workload remains constant, the speed-up is relatively good especially when using 1 processor per node. The computational intensive part of our application shows an insignificant improvement when we switch from one to two processors per node. On the other hand, the communication intensive portion shows a large improvement when switching from two to one processors per node. The reason for that is probably a less congested communication pattern.

Table 1.b shows the wall-clock for several scenarios (Fine, Coarse and two AMR runs) for one week of simulation. The application tuning for specific processor workload is a problem in itself, especially for parallel implementations, due to the difficulty in managing the amount of refinement that each processor does. Scenario AMR 1 is close to a quarter of the total fine wall-clock and close to our expectations in terms of accuracy as it will be shown below. AMR-2 is very competitive in terms of timing but the accuracy of the simulation is degraded (see Figs. 1.a and 2.a). In our experiments we noticed that accuracy is tightly linked to the number of mesh-points that are concentrated in the higher estimated truncation error locations.

The accuracy results are computed as the error mean for all 18 layers represented as error level contours. The error levels for O_3 after one week of simulation are shown in Figure 1.{a,b,c} for the two AMR results compared to the coarse simulation. The same results are also shown for the NO species in Figure 2.{a,b,c}. AMR-1 has a very high accuracy performance for both species, while AMR-2 has not performed so well. This suggests that insufficient refinement does not bring any significant gains in terms of accuracy. The mesh-point

Table 1. (a) - The wall-clock for one hour of simulated time on the finest refinement level when using one or two processors per node; (b) - Timing for fine, coarse and two AMR cases for one simulated week

No. of Procs.	Time [s] 1 proc./node	Time [s] 2 proc./node
16	2163	2739
32	1125	1270
64	841	1206
96	502	816

(a)

Simulation type	Time [s]	Final (Mean) no. of mesh-points
Fine	429,299	$36 \times 5058(5058)$
AMR-1	126,697	$36 \times 2310(2548)$
AMR-2	20,526	$36 \times 375(532)$
Coarse	4,150	$36 \times 150(150)$

(b)

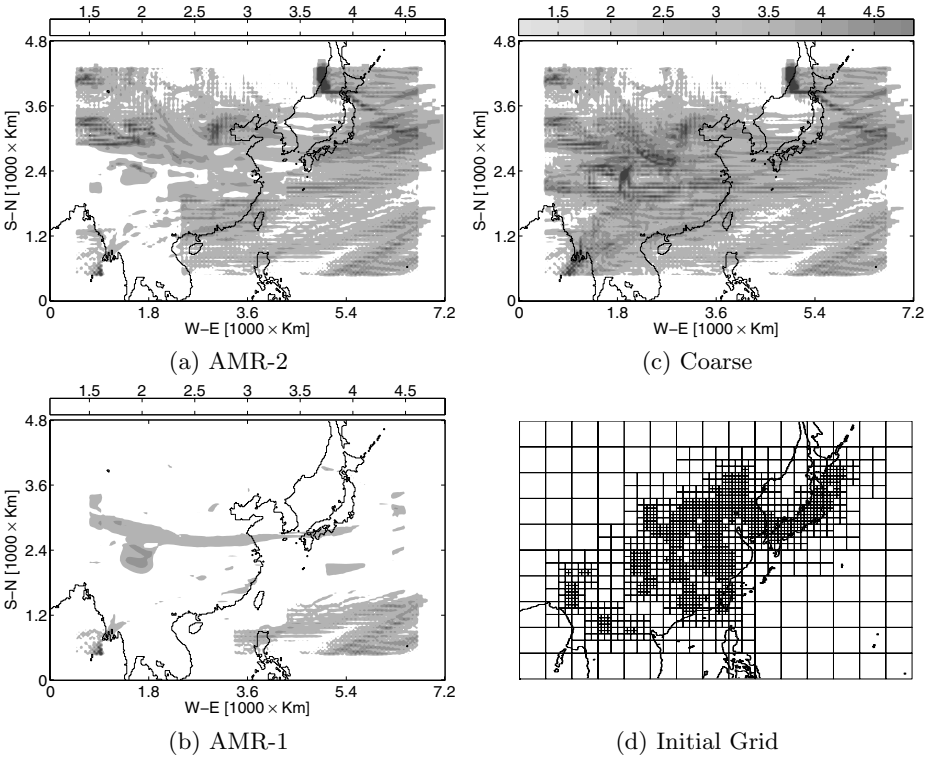


Fig. 1. Ozone error contours (percent) after one week of simulation ending at 0 GMT of March 11$^{\text{th}}$, 2001, for: (a) – AMR-2, (b) – AMR-1 and (c) – coarse level simulation. (d) – The refined grids at (0 GMT of March 1$^{\text{st}}$, 2001) for East Asia during the TraceP campaign. Each block (shown) consists of 6×6 computational cells (not shown). The criterion is based on the curvature of NO_X using $uptol = 0.25$ and $lowtol = 0.1$ with maximum refinement level 4 (10×10 Km)

dynamics over the one week period is shown in Figure 2.d. As we have expected, the system finds itself in a relative steady state - fine grids may move but the overall number of mesh-points is kept relatively constant, decreasing slowly as the solution becomes more and more smooth.

5 Conclusions

In this paper we investigate the parallel performance and accuracy improvements of an adaptive grid air pollution model based on the parallelized STEM air pollution model with the Paramesh tool. We look for accuracy improvements in O_3 and NO_X, species and low computational overheads.

The model scales very well as long as we use one processor per node. The communication intensive part plays a very important role as we tested our application

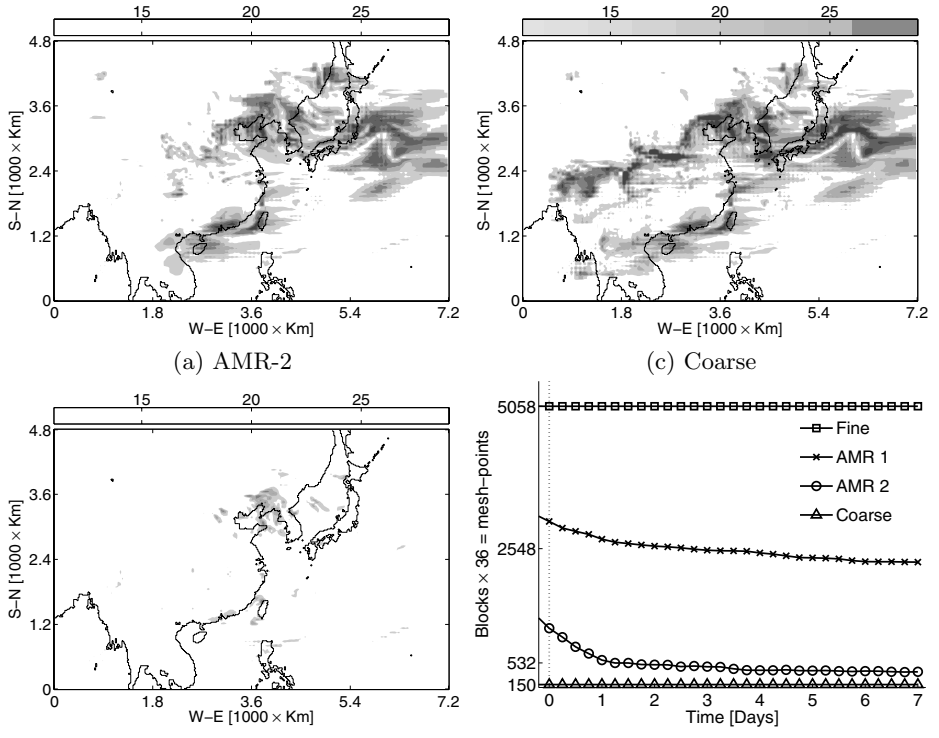


Fig. 2. NO error contours (percent) after one week of simulation ending at 0 GMT of March 11^{th}, 2001, for: (a) – AMR-2, (b) – AMR-1 and (c) – coarse level simulation. (d) – Represents the evolution of the number of blocks over the simulated week

on a network of workstations. This may be alleviated by a better mesh-point to processor partitioning.

The workload corresponding to the number of mesh-points is determined by the refinement criterion's refinement and derefinement thresholds. We found it very difficult to find appropriate values for those tolerances due to the relative autonomy of each process. A possible mitigation for this problem would be to collect all the mesh-point truncation errors, rank them and allow refinement/derefinement to a limited number of blocks. We chose to let the system evolve without any restrictions.

The accuracy benefits of AMR are amplified by the use of large number mesh-points: the finer the mesh, the better the accuracy. In our experiments, the use of almost triple the number of mesh-points than for the coarse simulation did not bring significant accuracy improvements to the final solution, while keeping the number of mesh-points between one quarter and one third of the fine one showed just a little solution degradation.

The dominant errors are located downwind of the emission sources especially for the high resolution AMR simulations (large number of mesh-points). A possible explanation is that the effect of errors in regions with high emissions are amplified by the chemical processes and advected downwind. This aspect would suggest a refined grid (increased resolution) upwind of the area of interest.

Acknowledgements

This work was supported by the National Science Foundation through the awards NSF CAREER ACI 0093139 and NSF ITR AP&IM 0205198. Our special thanks go to Virgina Tech's TCF for the use of the System X cluster.

References

1. C. Belwal, A. Sandu, and E. Constantinescu. Adaptive resolution modeling of regional air quality. *ACM Symposium on Applied Computing*, 1:235–239, 2004.
2. G.R. Carmichael. STEM – A second generation atmospheric chemical and transport model. *URL: http://www.cgrer.uiowa.edu*, 2003.
3. D.P. Chock, S.L. Winkler, and P. Sun. Effect of grid resolution and subgrid assumptions on the model prediction of non-homogeneous atmospheric chemistry. *The IMA volumes in mathematics and its applications: Atmospheric modeling, D.P. Chock and G.R. Carmichael editor*, pages 81–108, 2002.
4. G.R. Carmichael et. al. Regional-scale chemical transport modeling in support of the analysis of observations obtained during the TRACE-P experiment. *J. Geophys. Res.*, 108:10649–10671, 2004.
5. S. Ghorai, A.S. Tomlin, and M. Berzins. Resolution of pollutant concentrations in the boundary layer using a fully 3D adaptive technique. *Atmospheric Environment*, 34:2851–2863, 2000.
6. P. MacNeice and K. Olson. PARAMESH V2.0 – Parallel Adaptive Mesh Refinement. *URL: http://ct.gsfc.nasa.gov/paramesh/Users_manual/amr.html*, 2003.
7. P. MacNeice, K. Olson, and C. Mobarry. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126:330–354, 2000.
8. A. Sandu, Dacian N. Daescu, Gregory R. Carmichael, and Tianfeng Chai. Adjoint sensitivity analysis of regional air quality models. *Journal of Computational Physics*, :Accepted, 2004.
9. R.K. Srivastava, D.S. McRae, and M.T. Odman. Simulation of a reacting pollutant puff using an adaptive grid algorithm. *Journal of Geophysical Research*, 106(D20):24,245–24,257, 2001.
10. M. van Loon. *Numerical Methods in Smog Prediction.* Ph.D. Dissertation, CWI Amsterdam, 1996.

Total Energy Singular Vectors for Atmospheric Chemical Transport Models

Wenyuan Liao and Adrian Sandu

Department of Computer Science,
Virginia Polytechnic Institute and State University,
Blacksburg, VA 24061
{liao, asandu}@cs.vt.edu

Abstract. The aim of this paper is to address computational aspects of the total energy singular vector analysis of atmospheric chemical transport models. We discuss the symmetry of the tangent-linear/adjoint operator for stiff systems. Results for a 3D simulation with real data reveal that the total energy singular vectors depend on the target domain, simulation window, chemical reactions, and meteorological data.

Keywords: Adjoint models, sensitivity analysis, data assimilation, total energy singular vectors.

1 Introduction

Improvements of air quality require accurate and timely predictions of atmospheric pollutant concentrations. A critical element for accurate simulations is the use of observational data to constrain model predictions. Widely used data assimilation techniques include 3D-Var, 4D-Var, Kalman filter and ensemble nonlinear filters. Kalman filter techniques provide a stochastic approach to the data assimilation problem. The filter theory is described by Jazwinski [8] and the applicability to atmospheric modeling is discussed by Daley [4]. As explained by Fisher [6], the Kalman filter is too expensive to be a practical assimilation method for large-scale systems. The ensemble Kalman filter [7] is a feasible approach which approximates the Kalman filter covariance matrix by a Monte-Carlo-type technique.

In ensemble Kalman filters the random errors in the statistically-estimated covariance decrease only with the square-root of the ensemble size. Furthermore, the subspace spanned by the random vectors is not optimal for explaining the forecast error. For good statistical approximations with small size ensembles it is essential to properly place the initial ensemble to span the directions of maximum error growth. These directions are the total energy singular vectors as explained below. In this paper we study some of the challenges encountered when computing singular vectors for large transport-chemistry models.

The paper is organized as follows. In Section 2 we introduce the total energy singular vectors in the context of data assimilation. Computational aspects are discussed in Section 3, and numerical results are presented in Section 4. Conclusions and future directions are given in Section 5.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 806–813, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Ensembles and Singular Vectors

An atmospheric model propagates the model state (from $x_b(t_0)$ to $x_f(T)$) and its covariance matrix (from $P_b(t_0)$ to $P_f(T)$) using:

$$x_f = \mathcal{M}_{t_0 \to T}(x_b) \ , \quad P_f = M_{t_0 \to T} P_b M^*_{T \to t_0} + Q \ . \tag{1}$$

Here x_b and x_f represent the background and the forecast state, while P_b, P_f, and Q represent the covariance matrices of the errors in the background state, forecast state, and of the model errors respectively. The model solution operator is denoted by $\mathcal{M}_{t_0 \to T}$, $M_{t_0 \to T}$ is the solution operator of the tangent linear model and $M^*_{T \to t_0}$ the solution operator of its adjoint.

Consider a set of observables y (assumed, for simplicity, to be a linear function of model state, $y = Hx$). The extended Kalman filter uses the forecast state and its covariance $(x_f(T), P_f(T))$ and the observations and their covariance (y, R) to produce an optimal ("analyzed") estimation of the model state and its covariance $(x_a(T), P_a(T))$:

$$\begin{aligned} x_a &= x_f + P_f H^T (R + H P_f H^T)^{-1} (y - H x_f) \\ P_a &= P_f - P_f H^T (R + H P_f H^T)^{-1} H P_f \end{aligned} \tag{2}$$

The computational expense of the Kalman filter (2) is extremely large because one needs to invert the matrix $R + HP_fH^T$ and apply the tangent linear model to each column and the adjoint model to each row of the covariance matrix.

The commonly used method to reduce the computational cost is to propagate (only) the projection of the covariance matrix onto a low-dimensional subspace $\text{span}\{s_1, \cdots, s_k\}$. The subspace (at the analysis time T) should contain the directions $s_k(T)$ along which the error has the maximal growth.

Singular vector analysis was introduced in meteorology in the 60's by Lorenz [10] to compute the largest error growth rates. At the beginning of 90's, adjoint technique was introduced by Molteni [13] and Mureau [14] to compute singular vectors in meteorology problems, then singular vector analysis become viable with sophisticated atmospheric general circulation models (see e.g., Navon et. al. [15]).

We define the energy of an error vector at time t_0 as the Euclidean inner product $\langle s_k(t_0), A s_k(t_0) \rangle$, and the energy at the final time T as $\langle s_k(T), B s_k(T) \rangle$. A is a symmetric positive definite matrice and B is a symmetric positive semidefinite matrice. The errors evolve in time according to the dynamics of the tangent linear model, $s_k(T) = M_{t_0 \to T} s_k(t_0)$. The ratio between error energies at t_0 and T offers a measure of error growth:

$$\lambda = \frac{\langle s_k(T), B s_k(T) \rangle}{\langle s_k(t_0), A s_k(t_0) \rangle} = \frac{\langle s_k(t_0), M^*_{T \to t_0} B M_{t_0 \to T} s_k(t_0) \rangle}{\langle s_k(t_0), A s_k(t_0) \rangle} \tag{3}$$

The total energy singular vectors (TESV) are defined as the directions of maximal error growth, i.e. the vectors $s_k(t_0)$ that maximize the ratio λ in Eq.(3). These directions are the solutions of the generalized eigenvalue problem

$$M^*_{T \to t_0} B M_{t_0 \to T} s_k(t_0) = \lambda A s_k(t_0) \tag{4}$$

Eq.(4) can be solved efficiently using software packages like ARPACK [9] (or its parallel version PARPACK). The left side of Eq.(4) involves one integration with the tangent linear model followed by one integration with the adjoint model.

A special set of energy norms is provided by the choice $B = I$ and $A = (P_b)^{-1}$. In this case the resulting "Hessian" singular vectors $s_k(t_0)$ evolve into the leading eigenvectors $s_k(T)$ of the forecast error covariance matrix $P_f(T)$.

3 Computation of Chemical Singular Vectors

The numerical eigenvalue solver applied to (4) requires a symmetric matrix $M^* B M$ in order to successfully employ Lanczos iterations, and guarantee that the numerical eigenvalues are real. The symmetry requirement imposes to use the discrete adjoint M^* of the tangent linear operator M in (4). The computation of discrete adjoints for stiff systems is a nontrivial task [17]. In addition, computational errors (which can destroy symmetry) have to be small.

For a given model a symmetry indicator is constructed based on two random perturbation vectors $u(t_0)$ and $v(t_0)$ which are propagated forward in time, $u(\tau) = M_{t_0 \to \tau} u(t_0)$ and $v(\tau) = M_{t_0 \to \tau} v(t_0)$. The symmetry residual is the difference $r(\tau) = \langle u(\tau), M^*_{T \to \tau} M_{\tau \to T} v(\tau) \rangle - \langle v(\tau), M^*_{T \to \tau} M_{\tau \to T} u(\tau) \rangle$. Clearly if M^* is exactly the discrete adjoint of M then $r(\tau) = 0$ for all τ. However, both M and M^* are evaluated numerically and in practice we expect the symmetry residual $r(\tau)$ to have small (but nonzero) values.

As an example we consider the SAPRC-99 atmospheric gas-phase reaction mechanism [2] with 93 species and 235 reactions. The forward, tangent linear, and adjoint models are implemented using the automatic code generator KPP [3, 5, 17]. Several numerical experiments revealed that the magnitude of the symmetry residual depends on the choice of numerical integrator. Among the Rosenbrock integrators available in KPP Rodas4 [17] performs best. The variation of $r(\tau)$ with time for Rodas4 is shown in Fig. 1 (solid line). Surprisingly, the symmetry is lost for a short transient at the beginning of the time integration interval, where the symmetry residual jumps from 10^{-16} to 10^{-2}.

This behavior is due to the stiffness of the chemical terms. Consider a singular perturbation model for the chemical system $y' = f(y, z)$, $\epsilon\, z' = g(y, z)$. Here $\epsilon \ll 1$, y is the slow component, and z is the fast component. For $\epsilon \to 0$, the perturbation vectors that are propagated through the tangent linear model are of the form

$$\delta z = -g_z^{-1}(y, z) g_y(y, z) \delta y \tag{5}$$

During the numerical computation of the eigenvectors ARPACK (or any solver package) generates vectors $[\delta y, \quad \delta z]^T$ which do not satisfy Eq.(5).

To correct this we apply the tangent linear model on the initial perturbation for a short time, which is equivalent to 'projecting' the initial perturbation onto the slow evolution manifold described by (5). The result is then used to initialize the subsequent tangent linear model run. In order to preserve operator symmetry,

another projection using the adjoint model needs to be performed at the end of the adjoint integration. Consequently the operator is computed as

$$w = P^* M^*_{T \to t_0} M_{t_0 \to T} P\, u \ , \tag{6}$$

where P and P^* denote the projection operations performed with the tangent linear and the adjoint models respectively. Numerical tests revealed that a small number of projection steps (≤ 7) is sufficient in practice to substantially enhance symmetry.

Fig.1 (dashed) presents the evolution of the symmetry residual when 6 projection steps are performed with the very small stepsize of 10^{-9} seconds. The symmetry error during the transient is only 10^{-11}. □

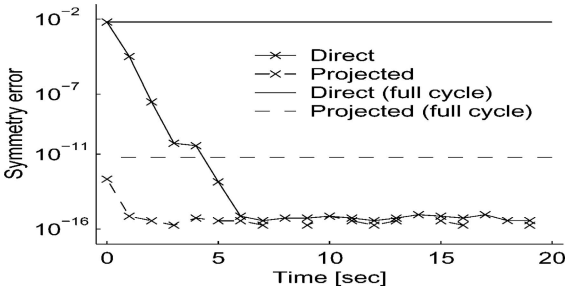


Fig. 1. Symmetry residual vs. time. Projection improves symmetry considerably

These results can be extended to 3D chemistry-transport models, which solve the advection-diffusion-reaction equations in the atmosphere. A detailed description of such models and the corresponding tangent linear and adjoint models is given in [16].

4 Numerical Results

The numerical tests use the state-of-the-art regional atmospheric chemical transport model STEM [1]. The simulation covers a region of 7200 Km × 4800 Km in East Asia and uses a $30 \times 20 \times 18$ computational grid with a horizontal resolution of 240 Km × 240 Km. The chemical mechanism is SAPRC-99 [2] which considers the gas-phase atmospheric reactions of volatile organic and nitrogen oxides in urban and regional settings. Both the forward and adjoint chemical models are implemented using KPP [3, 5, 17]. The simulated conditions correspond to March 2001. More details about the forward model simulation conditions and comparison with observations are available in [1]. The forward and adjoint models are parallelized using PAQMSG [12]. PARPACK [9] was used to solve the symmetric generalized eigenvalue problems.

To visualize the four-dimensional eigenvectors in (4) we consider separately the vector sections corresponding to different chemical species. Two-dimensional top views are obtained by adding the values in each vertical column.

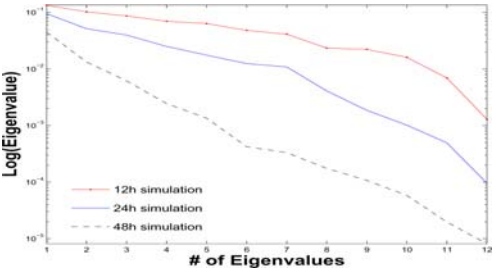


Fig. 2. The dominant eigenvalues for 12h, 24h and 48h simulations

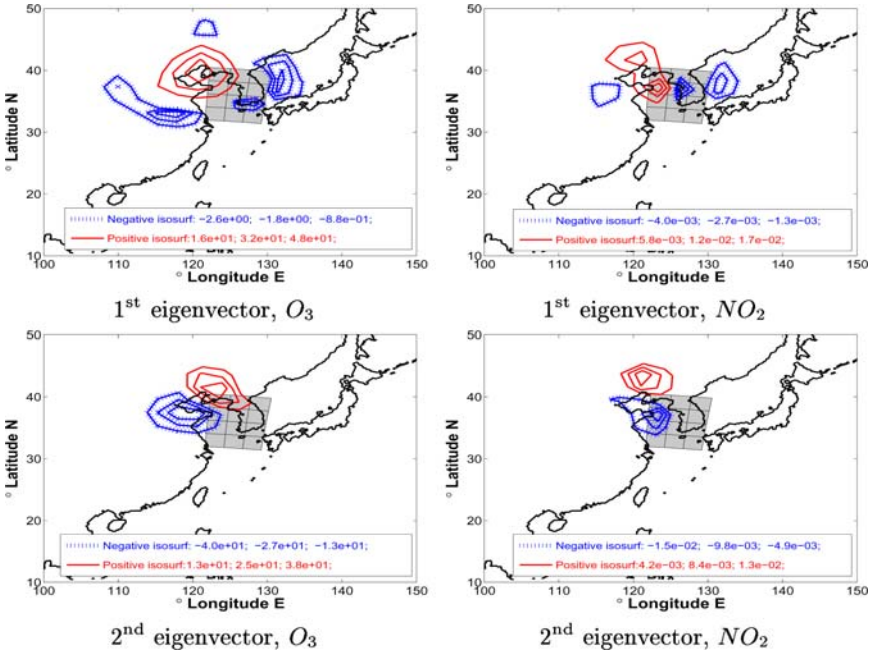


Fig. 3. Dominant eigenvectors for O_3 and NO_2, the 24h simulation

The target is the ground level ozone concentration in a 720 Km × 960 Km area covering Korea (the gray area in Fig. 3). The target (region, vertical level, and chemical species) defines the matrix B in (4).

The largest 12 eigenvalues for 12h, 24h and 48h simulations started at 0 GMT, March 1st, 2001 are shown in Fig. 2. The rapid decrease of eigenvalue magnitude indicates that one can capture the uncertainty in the target region with only a few total energy singular vectors. The eigenvalues decrease faster for longer simulation windows.

The O_3 and NO_2 sections of the first two dominant eigenvectors are shown in Fig. 3. The simulation interval for this test is 24 hours. We notice that the eigen-

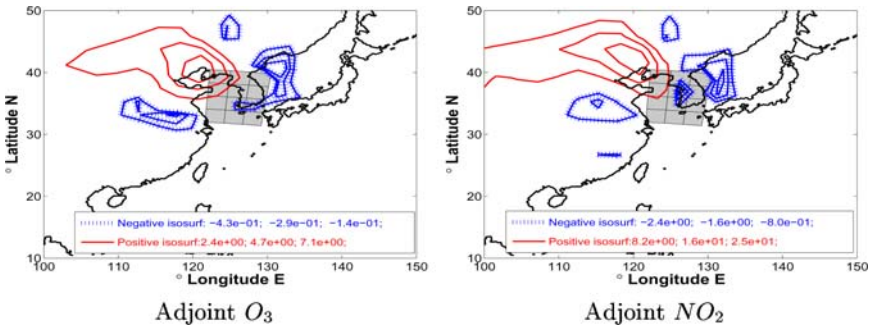


Fig. 4. Adjoint O_3 and NO_2 variables, the 24h simulation

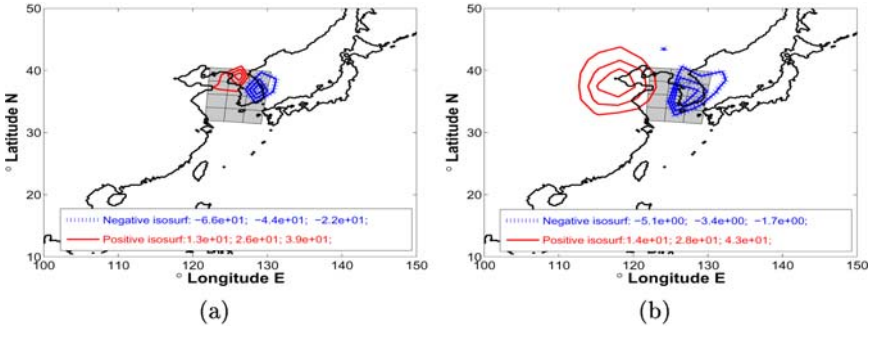


Fig. 5. Dominant O_3 eigenvectors for the 12h (a) and 48h (b) simulations

vectors are localized around the target area. The shapes of the second eigenvector is different from the first, which illustrates the fact that different eigenvectors contain different information. The shapes and the magnitudes of the O_3 and NO_2 sections are also different, illustrating the different influences that these species have on ground level O_3 after 24h.

Total energy singular vectors versus adjoints. To illustrate the difference between the information conveyed by the total energy singular vectors and adjoint variables we show the adjoints (for the total ground level O_3 in the target area after 24h) in Fig. 4. The adjoints cover a wider area following the flow pattern, while the singular vectors are more localized.

Influence of the simulation interval. The O_3 sections of the dominant eigenvectors for 12h and 48h simulations starting at 0 GMT, March 1, 2001, are shown in Fig. 5. The plots, together with Fig. 3, show the influence of the simulation interval on the singular vectors. For the 12h simulation the pattern is more localized.

Influence of meteorological conditions. The O_3 section of the dominant eigenvector for a 24h simulation started at 0GMT, March 26, 2001, is shown Fig. 6 (a). The shape of the TESV is different than for March 1st.

Influence of the target region. The O_3 section of the dominant eigenvector for another 24h, March 1st simulation is shown in Fig. 6(b). The target is ground

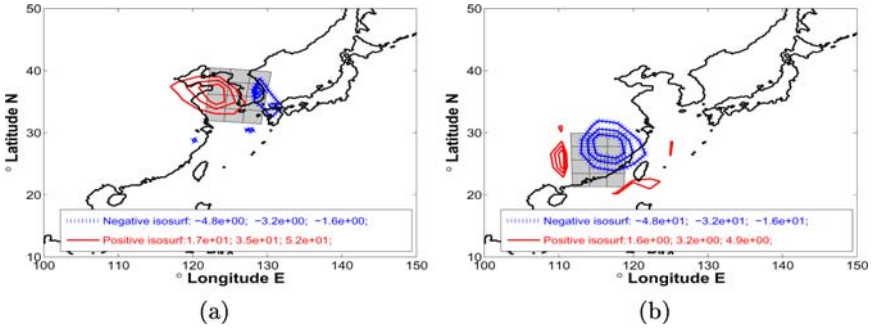


Fig. 6. Dominant eigenvectors (O_3 section) for: (a) Korea, March 26, show the influence of different meteorological conditions; and (b) China, March 1, show the effect of different target region

level ozone in a region of same area, but located in South-East China. Additional numerical tests revealed that the eigenvalues and eigenvectors are heavily effected by the size of target region. Specifically, the eigenvalues decrease is slower for larger regions, and therefore, more eigenvectors are needed to capture the uncertainty.

5 Conclusions

In this work we study the computational aspects of total energy singular vector analysis of chemical-transport models. Singular vectors span the directions of maximal error growth in a finite time, as measured by specific energy norms. The required symmetry of the tangent linear-adjoint operator implies the necessity of using discrete adjoints. A projection method is proposed to preserve symmetry for stiff systems associated with chemical models. Numerical results are presented for a full 3D chemistry-transport model with real-life data. The singular values/vectors depend on the simulation interval, meteorological data, location of target region, the size of target region etc. Future work will focus on computing Hessian singular vectors, and on using singular vectors within nonlinear ensemble filters.

Acknowledgements

This work was supported by the National Science Foundation through the awards NSF CAREER ACI 0093139 and NSF ITR AP&IM 0205198. We would like to thank Virginia Tech's laboratory for Advanced Scientific Computing (LASCA) for the use of the Anantham cluster.

References

1. Carmichael, G.R. et. al. "Regional-Scale Chemical Transport Modeling in Support of the Analysis of Observations obtained During the Trace-P Experiment". *Journal of Geophysical Research*, 108(D21), Art. No. 8823, 2004.

2. Carter, W.P.L. "Implementation of the SAPRC-99 chemical mechanism into the models-3 framework". *Technical Report*, United States Environmental Protection Agency, 2000.
3. Daescu, D., A. Sandu, G.R. Carmichael. "Direct and Adjoint Sensitivity Analysis of Chemical Kinetic Systems with KPP: II-Numerical Validation and Applications". *Atmospheric Environment*, **37**(36), 5097-5114, 2002.
4. Daley, R. *Atmospheric Data Analysis*. Cambridge University Press, 1991.
5. Damian,V, A. Sandu, M. Damian, F. Potra, G.R. Carmichael. "The Kinetic preprocessor KPP - a software environment for solving chemical kinetics". *Computers and Chemical Engineering*, **26**, 1567-1579, 2002.
6. Fisher, M. "Assimilation Techniques(5): Approximate Kalman filters and Singular Vectors", *Meteorological Training Course Lecture Seires*, 2001.
7. Houtekamer, P.L. and H.L. Mitchell. "A sequential Ensemble Kalman Filter for atmospheric data assimilation", *Monthly Weather Review* **129**, No. 1, 123-137, 2000.
8. Jazwinski, A.H. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.
9. Lehoucq, R., K. Maschhoff, D. Sorensen, C. Yang, ARPACK Software(Parallel and Serial), http://www.caam.rice.edu/software/ARPACK.
10. Lorenz, E.N. "A study of the predictability of a 28 variable atmospheric model". *Tellus*, **17**, 321-333, 1965.
11. Menut L., R. Vautard, M. Beekmann, C. Honor. "Sensitivity of photochemical pollution using the adjoint of a simplified chemistry-transport model". *Journal of Geophysical Research - Atmospheres*, **105-D12(15)**:15379-15402, 2000.
12. Miehe, P, A. Sandu, G.R. Carmichael, Y. Tang, D. Daescu. "A communication library for the parallelization of air quality models on structured grids". *Atmospheric Environment*, **36**, 3917-3930, 2002.
13. Molteni, F. and T.N. Palmer. "Predictability and finite-time instability of the northern winter circulation". *Quarterly Journal of the Royal Meteorological Society*, **119**, 269-298, 1993.
14. Mureau, R., F. Molteni, T.N. Palmer. "Ensemble prediction using dynamically-conditioned perturbations". *Quarterly Journal of the Royal Meteorological Society*, **119**, 299-323, 1993.
15. Li, Z., I.M. Navon, M.Y. Hussaini. "Analysis of the singular vectors of the full-physics FSU Global Spectral Model". *Tellus*, in press, 2005.
16. Sandu,A, D. Daescu, G.R. Carmichael, T. Chai. "Adjoint Sensitivity Analysis of Regional Air Quality Models". *Journal of Computational Physics*, in press, 2005.
17. Sandu,A, D. Daescu, G.R. Carmichael. "Direct and Adjoint Sensitivity Analysis of Chemical Kinetics Systems with KPP: I-Theory and Software Tools". *Atmospheric Environment*. **37**(36), 5083-5096, 2003.

Application of Static Adaptive Grid Techniques for Regional-Urban Multiscale Air Quality Modeling

Daewon Byun[1], Peter Percell[1], and Tanmay Basak[2]

[1] Institute for Multidimensional Air Quality Studies,
312 Science Research Bldg. University of Houston, Houston, Tx 77204-5007
{dwbyun, ppercell}@math.uh.edu
http://www.imaqs.uh.edu
[2] Department of Chemical Engineering, I.I.T. Madras, Chennai – 600 036, Inida
tanmay@iitm.ac.in

Abstract. Texas Air Quality Study 2000 revealed that ozone productivity in the Houston Ship Channel area was abnormally higher than other comparable cities in USA due to the large emissions of highly reactive unsaturated hydrocarbons from petrochemical industries. Simulations with popular Eulerian air quality models were shown to be inadequate to represent the transient high ozone events in the Houston Ship Channel area. In this study, we apply a multiscale Eulerian modeling approach, called CMAQ/SAFE, to reproduce the measured ozone productivity in the Houston Ship Channel and surrounding urban and rural areas. The modeling tool provides a paradigm for the multiple-level regional and local air quality forecasting operations that can utilize modern computational infrastructure such as grid computing technologies allowing to harness computing resources across sites by providing programmatic and high-bandwidth data linkage and establishing operational redundancy in the case of hardware or software failures at one operational site.

1 Introduction

Air quality in the Houston area suffers from high ozone levels. It is aggravated, by the considerable amounts of emissions of Volatile Organic Compounds (VOCs) from chemical processing plants distributed along the Houston Ship Channel and mobile NO_x (NO and NO_2) emissions from the traffics in the metropolitan area. In the presence of sunlight, VOCs and NO_x react in complex ways to form ozone. These concentrated local emissions (typically within sub-domains of size less than 4 km) pose a challenge for existing computational models such as the EPA Models-3 Community Multiscale Air Quality Modeling System (CMAQ) [1] in their current form. For example, during the recent TexAQS 2000 campaign, a wide variety of experimental measurements identified large and frequent Transient High Ozone Events (THOEs), i.e., spikes of ozone, that appear to be directly associated with releases of reactive unsaturated hydrocarbons near the Houston Ship Channel and in adjacent areas [2, 3]. On the other hand, simulations of air quality for the same period of time, using the existing emissions data and the highest resolution of the computational models, often fail to reproduce THOEs.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 814 – 821, 2005.
© Springer-Verlag Berlin Heidelberg 2005

CMAQ utilizes a regular grid approach that encounters difficulties in reproducing the ozone productivity measured in the Houston Ship Channel area during the TexAQS 2000 experiment, while a simpler Lagrangian model can be shown to successfully reproducing the observed behavior through the adjustment of input emissions, meteorological conditions, and back-ground chemical conditions. One approach to improve CMAQ is to increase the resolution of both horizontal advection and diffusion through the refinement of the underlying computational mesh, possibly irregular, quadrilateral mesh focused on a sub-domain of a regular rectangular CMAQ grid representing the density of emissions distribution. In the "Static Adaptive Fine-mesh Eulerian" (SAFE) modeling approach, we utilize the finite difference/volume algorithms on general quadrilateral meshes having the logical, but not necessarily geometric, structure of a regular rectangular grid. The algorithms now being used are extensions of ones that are already used in CMAQ with regular rectangular meshes. The advantages of utilizing SAFE grids are that they limit the computational expense of a highly refined grid to a user defined subdomain of interest and allow a variable degree of refinement inside the chosen subdomain.

2 Implementation Techniques

2.1 CMAQ Enhancements

The SAFE version of the CMAQ (CMAQ/SAFE) dynamically nests a subdomain with a refined SAFE quadrilateral grid inside of a coarser standard grid called the parent grid. Computations proceed in lock step on both the parent and SAFE grids, with the SAFE grid dynamically receiving interpolated boundary concentrations for tracked species from the latest results on the parent grid. This is dynamic one-way nesting. Normal nesting of CMAQ grids, say a 4km grid inside of a 12-km grid, is done with separate executions of CCTM, first on the 12-km grid and then, after a boundary condition file has been generated from the results on the 12km grid, the run on the 4km grid reads in the boundary condition file. CMAQ/SAFE does dynamic nesting because a future goal is to do two-way nesting, with the run on the refined SAFE grid dynamically feeding its results back to the parent grid. When no initial conditions are available on the SAFE grid, the initial conditions are simply interpolated in space from the initial conditions being used on the parent grid.

To be able to run on a SAFE mesh, "SAFE aware" versions of a number of process subroutines were created. In many cases this just required that the subroutine be modified to process the list of cells in the SAFE grid, rather than in the parent grid. The following briefly describes a list of CMAQ processes modified to create a SAFE version of the associated code:

- Reading External Meteorological and Emission Inputs: Data are read at the parent grid resolution and then interpolated in space to the SAFE grid to provide dynamic boundary conditions.
- Horizontal Mass Transport Processes: Horizontal advection and diffusion algorithms were modified to be "SAFE aware"

- Vertical Mass Transport Processes: No modification was needed for the vertical advection and diffusion processes because the same horizontal SAFE mesh is used at all vertical layers
- Single Cell Processes: Processes such as Chemistry required modifications to aware the new SAFE data structures.

Other processes, such as Plume-in-Grid modeling, Cloud processes and Aerosol modeling, that are not as significant in modeling ozone production, are not yet supported on SAFE grids.

2.2 Grid Structures

The CMAQ/SAFE code and data structures have been designed to work with any horizontal mesh of non-degenerate convex quadrilateral cells that reversibly maps to a regular rectangular grid. This restriction permits use of techniques such as operator splitting that have traditionally been used in air quality modeling on uniform rectangular grids for the irregular grid system. It also simplifies data management tasks and CMAQ's present rectangular data infrastructure based on I/O API can be utilized as is. In principle, if a user supplied the positions of all the nodes of any such a grid, then that data could easily be read into the SAFE data structures and used for a simulation.

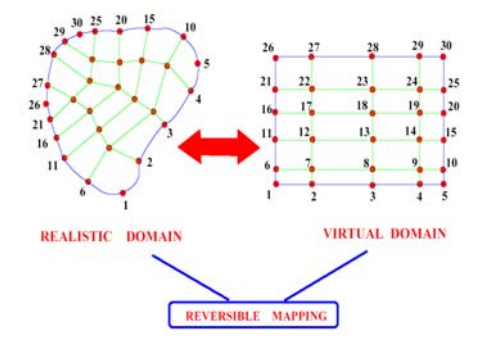


Fig. 1. Demonstration of the structural equivalence of a SAFE grid on an irregular domain with a regular rectangular grid

At this point, a few simple static grid generation algorithms are available as options through CMAQ/SAFE. These algorithms require a very small amount of input data to define the SAFE subdomain and its mesh in terms of the rows and columns of the parent grid. A good deal of work has been done on grid generation algorithms, especially algorithms that automatically adapt the mesh to particular problem and solution features. Investigating such algorithms and incorporating them into CMAQ is a possible area for future work.

2.3 Horizontal Transport on a SAFE Mesh

The governing conservation equation for trace species advection is

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{V}) = 0 \tag{1}$$

where ϕ is the concentration (as a volume density) of a trace species, and $\mathbf{V}$ is the velocity vector of the wind (again, after transformation to the computing coordinate system). The solution method is the same for both air and the trace species, so we shall present the approach used in terms of a trace species. In order to discretize the advection equation on a SAFE mesh, we consider each quadrilateral SAFE cell to be a control volume. With the approximations that ϕ is constant on each cell and that the velocity vector $\mathbf{V}$ is constant along each cell edge, over the duration of a time step Δt we get the equation

$$\frac{\Delta(A\phi)}{\Delta t} = -(L_E \phi \boldsymbol{\nu}_E \cdot \mathbf{V}_E - L_W \phi \boldsymbol{\nu}_W \cdot \mathbf{V}_W) - (L_N \phi \boldsymbol{\nu}_N \cdot \mathbf{V}_N - L_S \phi \boldsymbol{\nu}_S \cdot \mathbf{V}_S) \tag{2}$$

where A is the area of the cell, L is the length of an edge and $\boldsymbol{\nu}$ is the unit normal vector to an edge, as shown in Figure 2.

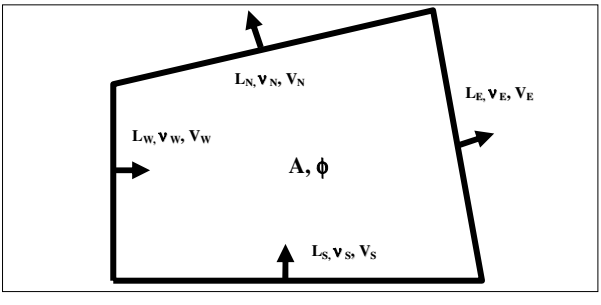


Fig. 2. A typical quadrilateral cell in a SAFE mesh. The arrows show the direction of the unit normal vectors $\boldsymbol{\nu}$

By rearranging Equation (2), and splitting it into the (nominal) West-East and South-North directions, we get

$$\begin{aligned} \Delta\phi\big|_{EW} &= -\frac{\Delta t(L_E \phi \boldsymbol{\nu}_E \cdot \mathbf{V}_E - L_W \phi \boldsymbol{\nu}_W \cdot \mathbf{V}_W)}{A} \\ \Delta\phi\big|_{NS} &= -\frac{\Delta t(L_N \phi\big|_{EW} \boldsymbol{\nu}_N \cdot \mathbf{V}_N - L_S \phi\big|_{EW} \boldsymbol{\nu}_S \cdot \mathbf{V}_S)}{A} \end{aligned} \tag{3}$$

With this splitting we can advance the advection solution a time step by sending the first piece for each row of quadrilaterals and the second piece for each column of quadrilaterals to a solver for one dimensional advection. This is done by telling the solver that $L\mathbf{v}\cdot\mathbf{V}$ is a speed and A is a "distance" step. The solver has the job of finding a good approximation to the average value of ϕ for the area $(L\mathbf{v}\cdot\mathbf{V})\Delta t$ that flows through an edge of the cell over a time step. Then $\phi_{ave}L\mathbf{v}\cdot\mathbf{V}\Delta t$ is the total mass that flows through the edge over the time step. Note that this formulation is guaranteed to conserve mass, but the amount of mass movement between cells is subject to approximation error.
The equation for horizontal diffusion is

$$\frac{\partial \phi}{\partial t} = \nabla \cdot \left(\rho_J K_H \nabla q\right) \tag{4}$$

where ρ_J is the density of air times the Jacobian determinant of a coordinate transformation, K_H is a scalar eddy diffusivity dependent on wind deformation and $q = \phi / \rho_J$ is mixing ratio. In CMAQ/SAFE this equation is now handled like the advection equation, with the mass flux $\rho_J K_H \nabla q$ replacing $\phi\mathbf{V}$. In fact, it is sufficient to use the resulting version of Equation 0 directly, without operator splitting and a higher order approximation method as in the one dimensional advection solver being used.

3 CMAQ/SAFE Simulation Results

The domain for our studies of CMAQ/SAFE is shown in Figure 3. This is a domain that contains Houston, the Houston ship channel, Galveston and surrounding areas. For a parent grid with 4km cells, the cells in the skewed mesh average out to 2km cells, while the smallest cells in the other mesh are 1km and the cells in the corner areas are 2km cells. Users can choose arbitrary boundary nodes following an irregularly shaped enclosure to adapt the grid to a non-rectangular domain of interest.

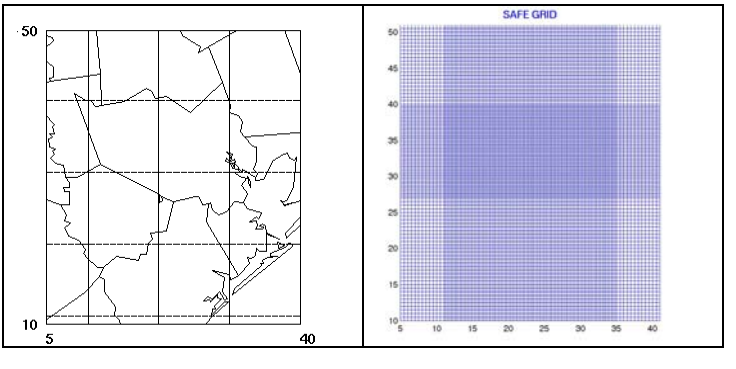


Fig. 3. Subdomain and SAFE mesh used for comparing selected horizontal and vertical results on the parent grid and SAFE grid

3.1 Transport Process of CMAQ/SAFE

Here, we simulated transport processes only, i.e., horizontal and vertical advection and diffusion, using non-reactive tracer species with artificially controlled initial and boundary conditions. The test validates the use of SAFE meshes and the modifications of the advection and diffusion solution algorithms. We compared results (not shown here) for the small subdomains that were obtained with the parent grid's 4km mesh, the irregular rectangular SAFE mesh, and the skewed SAFE mesh. The study shows good agreement between the results on the three different grids, with the results on the SAFE grids obviously showing more pronounced local variation. All 3 meshes are coarse enough so that we cannot expect exact agreement. An important thing to notice is that even though the two SAFE grids are distinctly different, the differences in the computed concentrations are quite small. Clearly the choice of mesh, at this degree of refinement, does not change the qualitative results, but does give slightly different local variation in concentrations, which is to be expected.

3.2 Effects of SAFE Mesh on Nonlinear Chemistry

We have tested CMAQ/SAFE with transport, emissions and the SAPRC99 gas-phase chemistry mechanism. The parent grid has a 4-km mesh and the SAFE grid used here has a variable rectangular mesh with cell edges either 1-km or 2-km (see Figure 3). It contains 120 columns and 108 rows or 12,960 cells. The grid resolution affects the simulated ozone production because it is produced through the nonlinear chemical reactions of the VOC and NO_x species. When the emissions of either VOC or NO_x species are dispersed in a too larger cell and fail to represent the actual atmospheric conditions, the model would not be successful in generating observed ozone concentrations.

Figure 4 shows NO_x concentration peaks in the Houston downtown area, with high volume of traffics, thus large NO_x emissions. SAFE grid provides a more detailed distribution of NO_x. However, because the road network is wide spread over the large downtown area, there is not much difference in the peak values. On the other hand, when highly reactive VOC species are emitted in a small but concentrated industrial area, such as the Houston Ship Channel, the model grid should be sufficiently small to properly describe NO_x-VOC concentration ratios to replicate real atmospheric conditions. Figure 4 also contrasts the resulting ozone concentration differences. The CMAQ/SAFE with higher grid resolution can successfully simulate the high ozone event in the Houston. It is evident in Figure 5 that the high ozone concentration from the CMAQ/SAFE resulted from the higher HCHO concentrations, which in turn was produced through the photo-oxidation of the highly reactive VOC emissions from the Ship Channel industries. Aircraft measurements during the TexAQS studied showed the presence of high HCHO and ozone plumes in around and downwind of the Houston Ship Channel area. Only when the NO_x-VOC concentration ratios are properly represented, the model can simulate the high ozone concentrations observed.

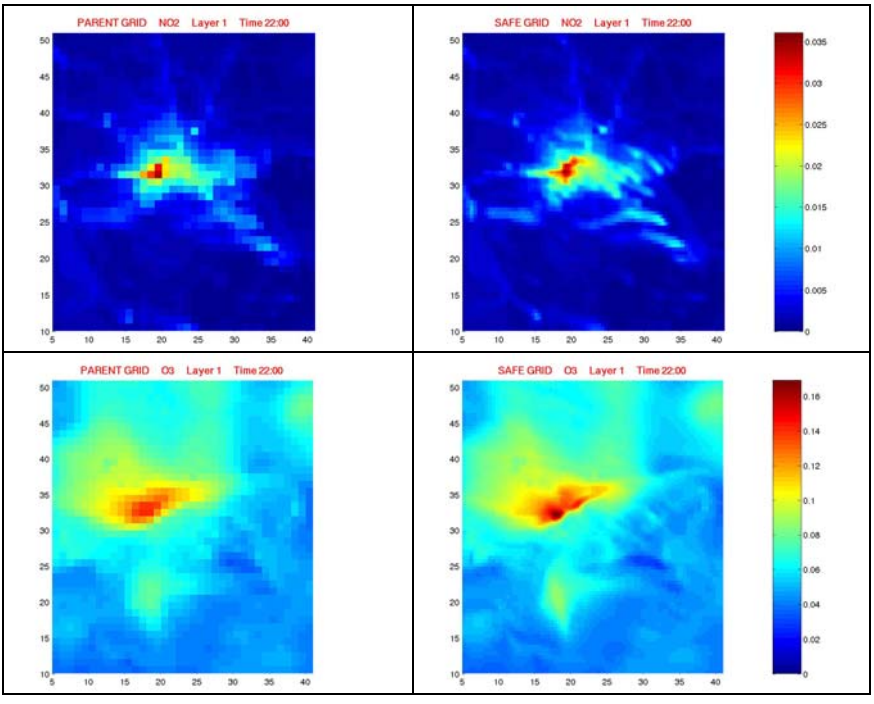


Fig. 4. Comparison of NO_2 (top) and O_3 (bottom) concentrations between the parent grid (left) and SAFE grid (right)

4 Conclusions

The goal of this study was to implement CMAQ the capability to efficiently focus computing resources for enhanced resolution of its science processes in geographical areas where the user needs an extra degree of detail. The SAFE grid in the target sub-window is a more refined and more flexible grid, e.g., with rectangular, or even quadrilateral cells, whose shape and size can vary within the grid. The solution on the SAFE grid provides more detail inside its sub-window, but currently does not improve the solution on the parent grid. At this time, many of the science processes have been extended to work simultaneously on a regular "parent" grid and a SAFE grid on a single targeted sub-window. Further enhancement tasks under development are; (1) two-way nesting solution to utilize the refined sub-grid results feed back to the parent grid, and (2) multiple SAFE domains inside the same parent grid. For example, this capability could be used to improve modeling of the air quality interaction between two large, but separate, cities, such as Houston and Dallas within Texas. The two-way nesting capability described above is needed to realize an improvement in the interaction between improved solutions in multiple SAFE windows.

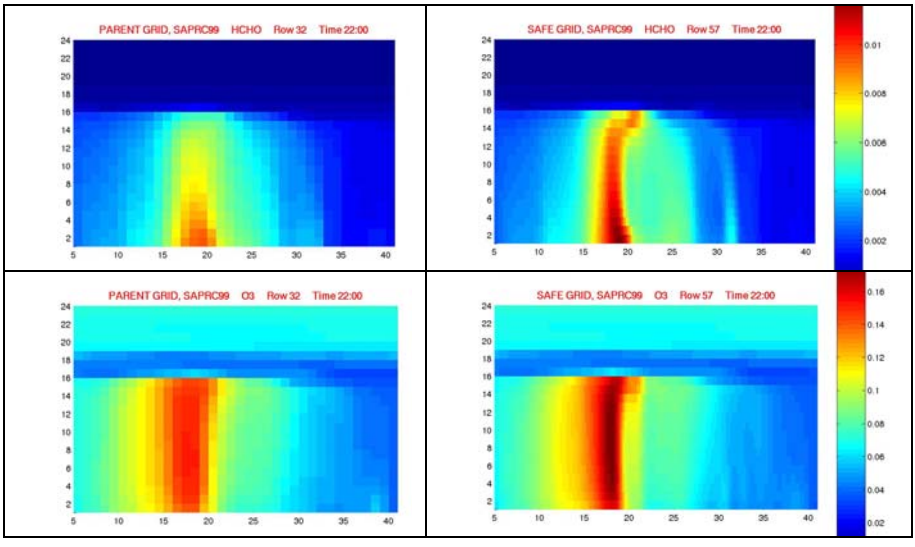


Fig. 5. Vertical cross section over the Ship Channel area of HCHO (top) and O_3 (bottom) for the parent mesh (left) and the SAFE mesh (right)

References

1. Byun, D.W. and Ching, J.K.S.: Science algorithms of the EPA Models-3 Community Multiscale Air Quality (CMAQ) Modeling System". EPA-600/R-99/030, U.S. EPA. (1999) (available at http://www.epa.gov/asmdnerl/models3/doc/science/science.html)
2. Byun, D.W., Kim, S.-T., Cheng, F.-Y., Kim, S.-B., Cuclis, A., and Moon, N.-K.: Information Infrastructure for Air Quality Modeling and Analysis: Application to the Houston-Galveston Ozone Non-attainment Area, *J. Environmental Informatics*, **2**(2) (2003) 38-57
3. Daum, P.H., L. I. Kleinman, S. R. Springston, L. J. Nunnermacker, Y.-N. Lee, J. Weinstein-Lloyd, J. Zheng, and C. M. Berkowitz: A comparative study of O3 formation in the Houston urban and industrial plumes during the 2000 Texas Air Quality Study. *Journal of Geophysical Research*, **108**(D23) (2003) 4715, doi:10.1029/2003JD003552.

On the Accuracy of High-Order Finite Elements in Curvilinear Coordinates

Stephen J. Thomas and Amik St.-Cyr

National Center for Atmospheric Research,
1850 Table Mesa Drive, Boulder, 80305 CO, USA
{thomas, amik}@ucar.edu

Abstract. The governing equations for shallow water flow on the sphere are formulated in generalized curvilinear coordinates. The various analytic expressions for the differential operators are all mathematically equivalent. However, numerical approximations are not equally effective. The accuracy of high-order finite element discretizations are evaluated using the standard test problems proposed by Williamson et al (1992). The so-called strong conservation formulation is far more accurate and results in standard error metrics that are at least two orders of magnitude smaller than the weak conservation form, Jorgensen (2003), Prusa and Smolarkeiwicz (2003). Moreover, steady state solutions can be integrated much longer without filtering when time-stepping the physical velocities.

1 Introduction

The various terms that arise in a numerical model of the atmospheric general circulation (e.g. material derivative, gradient and divergence) have a tangible, physical existence that is independent of any coordinate-based description. However, coordinate-based representations are necessary for computing the explicit form of all requisite terms. Because the precise form of these terms depends upon the coordinate system being employed, a tensor representation is preferable. It allows the use of powerful mathematical tools to deduce relations that are valid in any coordinate system, i.e. coordinate invariant forms, while conveying the physical interpretation of the symbolic representation. For example, the equations of motion can be formulated using four different forms of the velocity; physical, contravariant, covariant or solenoidal. Although analytically equivalent, these lead to numerical approximations that are not equally effective. Here, we consider high-order finite element discretizations of the governing equations in curvilinear coordinates.

High-order finite elements are well-suited to atmospheric modeling due to their desirable numerical properties and inherent parallelism. The development of discontinuous Galerkin approximations can be viewed as an extension of low order finite-volume techniques for compressible flows with shocks (Cockburn et al 2000). Either nodal or modal basis functions can be employed in high-order

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 822–828, 2005.
© Springer-Verlag Berlin Heidelberg 2005

finite elements which are spectrally accurate for smooth solutions. A discontinuous Galerkin approximation is equivalent to a conservative finite-volume discretization where the jump discontinuity at an element boundary requires the solution of a Riemann problem. In general, a filter is required to stabilize long time integrations of the nodal discontinuous approximation of the shallow water equations, due to the presence of nonlinear terms and their integration using inexact Gaussian quadrature, Giraldo et al (2003).

2 Shallow Water Equations

The shallow water equations contain the essential wave propagation mechanisms found in atmospheric general circulation models. These are the fast-moving gravity waves and nonlinear Rossby waves. The flux form shallow-water equations in curvilinear coordinates on the cubed-sphere are described in Rancic et al (1996) and Sadourny (1972). Covariant and contravariant vectors are related through the metric tensor by $u_i = G_{ij}u^j$, $u^i = G^{ij}u_j$, and $G = \{\det(G_{ij})\}^{1/2}$.

$$\begin{aligned} \frac{\partial u_1}{\partial t} &= -\frac{\partial}{\partial x^1}E + G\,u^2(f+\zeta) \\ \frac{\partial u_2}{\partial t} &= -\frac{\partial}{\partial x^2}E - G\,u^1(f+\zeta) \\ \frac{\partial}{\partial t}(G\,\Phi) &= -\frac{\partial}{\partial x^1}(G\,u^1\Phi) - \frac{\partial}{\partial x^2}(G\,u^2\Phi) \end{aligned} \tag{1}$$

where

$$E = \Phi + \frac{1}{2}\,(u_1\,u^1 + u_2\,u^2), \quad \zeta = \frac{1}{G}\left[\frac{\partial u_2}{\partial x^1} - \frac{\partial u_1}{\partial x^2}\right].$$

h is the height above sea level. u^i and u_j are the contravariant and covariant velocities. $\Phi = gh$ the geopotential height. f is the Coriolis parameter.

The metric tensor for all six faces of the cube is

$$G_{ij} = \frac{1}{r^4\cos^2 x^1\,\cos^2 x^2}\begin{bmatrix} 1+\tan^2 x^1 & -\tan x^1\,\tan x^2 \\ -\tan x^1\,\tan x^2 & 1+\tan^2 x^2 \end{bmatrix}$$

where $r = (1+\tan^2 x^1 + \tan^2 x^2)^{1/2}$ and $G = 1/r^3\cos^2 x^1\,\cos^2 x^2$. A vector $\mathbf{v} = (v_1, v_2)$ in spherical coordinates is defined by its covariant and contravariant components. For the vector (u^1, u^2) on the cube, the Jacobi matrix of the transformation between the sphere and cube is given by

$$D = \begin{bmatrix} \cos\theta\,\partial\lambda/\partial x_1 & \cos\theta\,\partial\lambda/\partial x_2 \\ \partial\theta/\partial x_1 & \partial\theta/\partial x_2 \end{bmatrix},$$

where $D^T D = G_{ij}$ and the cube to sphere mappings are

$$D^T\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad D\begin{bmatrix} u^1 \\ u^2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.$$

The system of equations (1) is written in a weak conservation form. An alternative approach is to use a strong conservation form with physical velocities on the lhs of the momentum equation.

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} &= -(f+\zeta)\,\mathbf{k}\times\mathbf{v} - \frac{1}{2}\nabla(\mathbf{v}\cdot\mathbf{v}) - \nabla\Phi \\ \frac{\partial \Phi}{\partial t} &= -\nabla\cdot\Phi\mathbf{v}. \end{aligned} \tag{2}$$

where

$$\nabla\Phi = D^{-T}\left[\frac{\partial\Phi}{\partial x^1},\ \frac{\partial\Phi}{\partial x^2}\right]^T, \quad \nabla\cdot\Phi\mathbf{v} = \frac{1}{G}\left[\frac{\partial G\,u_1\Phi}{\partial x^1} + \frac{\partial G\,u_2\Phi}{\partial x^2}\right].$$

The kinetic energy is coordinate invariant and expressed in terms of spherical velocity components. The gradient is a covariant quantity and is mapped back to physical coordinates by the Jacobi matrix. The divergence is a scalar obtained by differentiating the contravariant velocity components. Taylor et al (1997) use an alternative approach where the time derivatives of the contravariant velocity components appear on the lhs of the momentum equation. Our numerical experiments clearly demonstrate that time integration of (2) leads to much smaller errors and reduces the amount of filtering required to stabilize the scheme.

3 Space Discretization

The computational domain Ω is partitioned into finite elements Ω_k. An approximate solution u_h belongs to the finite dimensional space $\mathcal{V}_h(\Omega)$. u_h is expanded in terms of a tensor-product of the Lagrange basis functions defined at the Gauss-Lobatto-Legendre points

$$u_h^k = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij} h_i(x) h_j(y)$$

A weak Galerkin variational problem is obtained by integrating the equations with respect to a test function $\varphi_h \in \mathcal{V}_h$. In the continuous Galerkin spectral element method, integrals are evaluated directly using Gauss-Lobatto quadrature

$$\int_{\Omega_k} \phi_h u_h \, d\Omega = \sum_{i=0}^{N}\sum_{j=0}^{N} \phi_h(\xi_i,\xi_j) u_h(\xi_i,\xi_j)\rho_i\rho_j \tag{3}$$

where (ξ_i, ρ_i) are the Gauss-Lobatto nodes and weights. C^0 continuity is imposed in the spectral element method through the application of direct stiffness summation, Deville et al (2002).

To illustrate the discontinuous Galerkin approach, consider a scalar hyperbolic equation in flux form,

$$u_t + \nabla\cdot\mathcal{F} = S.$$

By applying the Gauss divergence theorem, the weak form becomes

$$\frac{d}{dt}\int_{\Omega_k}\varphi_h u_h\,d\Omega = \int_{\Omega_k}\varphi_h S\,d\Omega + \int_{\Omega_k}\mathcal{F}\cdot\nabla\varphi_h\,d\Omega - \int_{\partial\Omega_k}\varphi_h\mathcal{F}\cdot\hat{n}\,ds$$

The jump discontinuity at an element boundary requires the solution of a Riemann problem where the flux function $\mathcal{F}\cdot\hat{n}$ is approximated by a Lax-Friedrichs numerical flux. The resulting semi-discrete equation is given by

$$\frac{du_h}{dt} = L(u_h).$$

4 Numerical Experiments

Numerical experiments are based on of Williamson et al (1992). Test case 2 is a stationary zonal geostrophic flow. In theory, the equations can be integrated indefinitely with the only source of errors being due to numerical truncation. The continuous and discontinuous Galerkin spectral element models were integrated over 300 days using both the weak (1) and strong (2) conservation forms of the shallow water equations. The total number of elements was $9 \times 6 = 54$ and the number of Gauss-Lobatto-Legendre points per element was set to 16×16. In the case of the continuous Galerkin model, a second order explicit leap frog time integration scheme is applied together with a Robert-Asselin (1972) time filter to damp the resulting computational mode. The time step size was $\Delta t = 30$ sec, respecting the CFL condition. The discontinuous Galerkin model is integrated with the second order three stage SSP Runge-Kutta scheme of Higueras (2004) using the same time step size. A spatial filter to stabilize the time step was not applied during any of these integrations, Fischer and Mullen (2001). Figure 1 contains plots of the eigenvalues of the inverse metric tensor G^{ij} used to map covariant to contravariant quantities on the cube. These clearly illustrate the magnitude of the stretching across one face of the cube-sphere.

The results of the continuous Galerkin model integrations reveal that the weak formulation becomes unstable after only six days. Whereas the strong form can be stably integrated for over 300 days without any spatial filtering with a small growth in the error level. We attribute these results to several factors. Weak conservative formulations implicitly map between the sphere and cube with the inverse metric tensor, whereas the strong form relies on the Jacobi matrix. The eigenvalues of the latter are a factor of $\sqrt{2}$ smaller. Figure 2 is a plot of the l_∞ geopotential height errors for the continuous Galerkin formulations. These plots show that the errors are two orders of magnitude smaller for the strong conservation formulation. The results for the discontinuous Galerkin scheme are similar, namely the weak formulation becomes unstable after six days of integration. Once again the strong form remains stable for well over 300 days of integration. Furthermore, the scheme conserves mass and the l_∞ error remains close to machine precision.

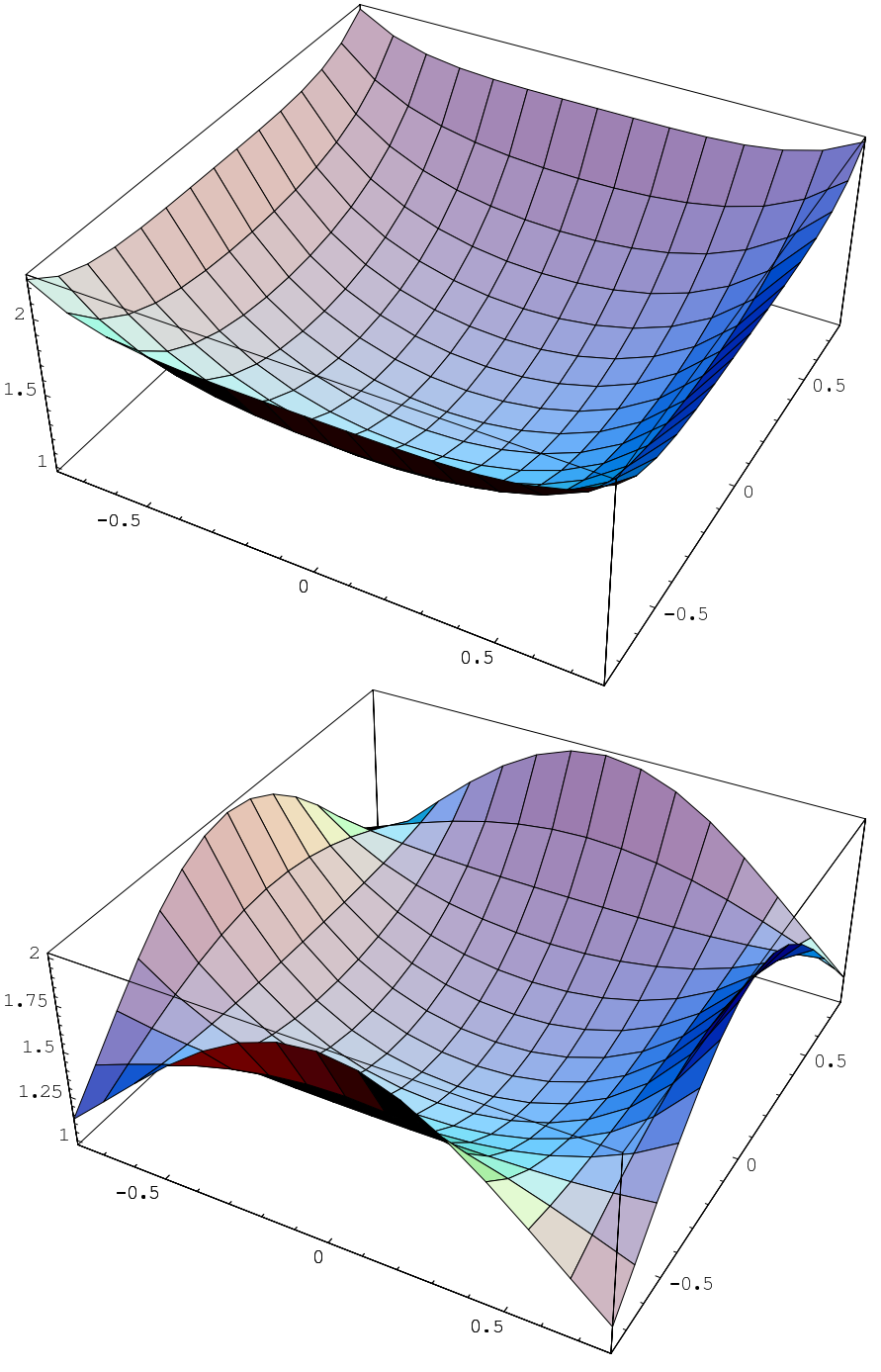


Fig. 1. Eigenvalues of the inverse metric tensor G^{ij} used to map covariant to contravariant quantities on the cube. Top: λ_1. Bottom: λ_2

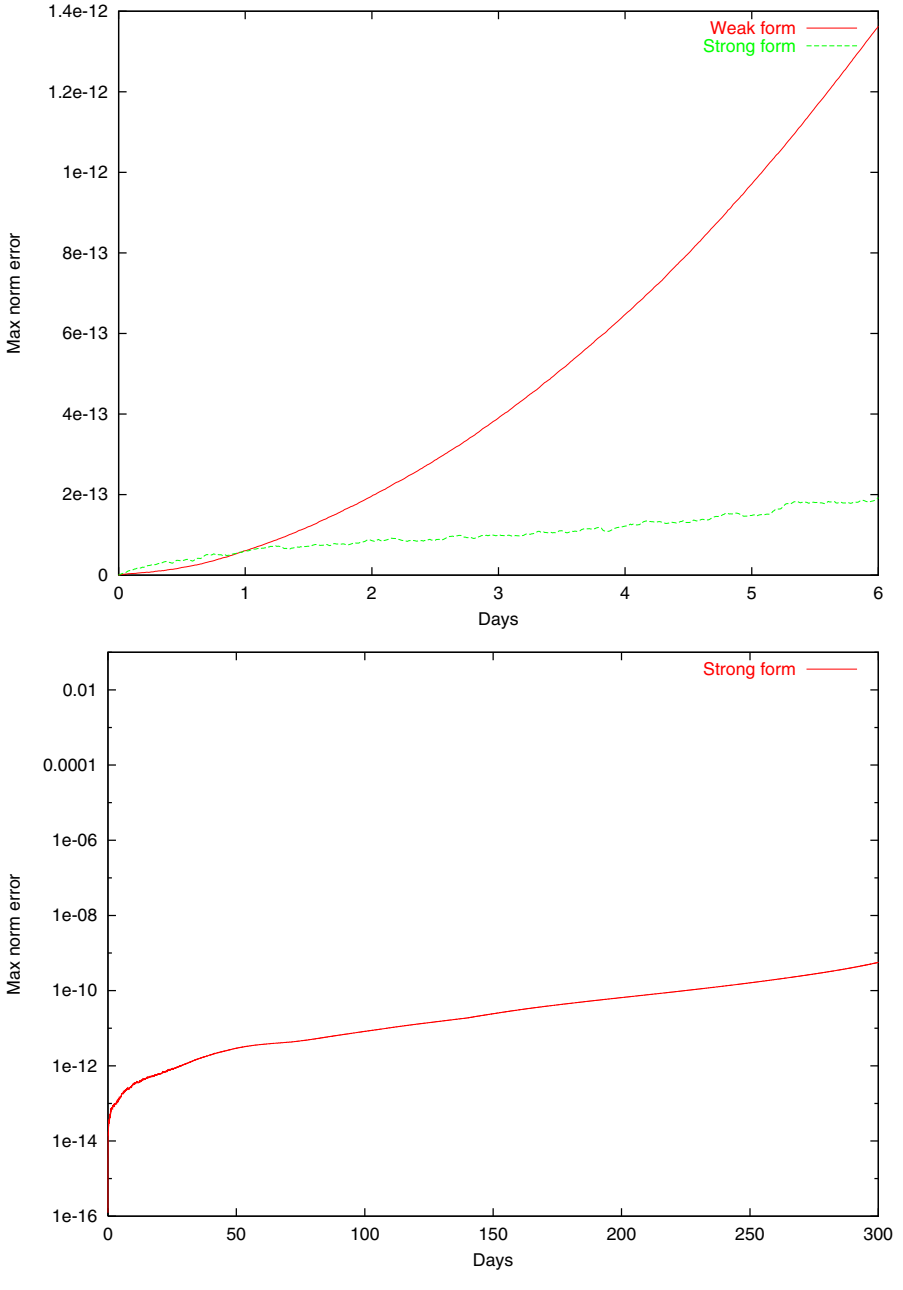


Fig. 2. Shallow water test case 2: Stationary geostrophic flow. $9 \times 6 = 54$ continuous Galerkin spectral elements. 16×16 Gauss-Lobatto-Legendre points per element. $\Delta t = 30$ sec. No spatial filter applied. Top: l_∞ error for weak and strong conservation forms integrated to 6 days. Bottom: l_∞ for strong conservation form integrated to 300 days

5 Conclusions

Taylor et al (1997) were not able to achieve or maintain machine precision level errors because they employed a weak conservation form for the governing equations of motion. To fully exploit the accuracy of high-order finite elements, a judicious choice of an appropriate form of the prognostic equations is required when using generalized curvilinear coordinates. Our experience with a nodal Galerkin method indicates that a filter is ultimately required for long integrations to stabilize the scheme, thereby improving and extending the recent results of Nair et al (2004).

References

1. Asselin, R., 1972: Frequency filter for time integrations. *Mon. Wea. Rev.*, **100**, 487–490.
2. Cockburn, B., G. E. Karniadakis, and C. W. Shu, 2000: Discontinuous Galerkin Methods. Springer-Verlag, New York, 470 pp.
3. Deville, M. O., P. F. Fischer, and E. H. Mund, 2002: High-Order Methods for Incompressible Fluid Flow. Cambridge University Press, 499 pp.
4. Fischer, P. F., and J. S. Mullen, 2001: Filter-Based stabilization of spectral element methods. *Comptes Rendus de l'Académie des sciences Paris*, t. 332, Série I - Analyse numérique, 265–270.
5. Giraldo, F. X., J. S. Hesthaven, and T. Warburton, 2003: Nodal high-order discontinuous Galerkin methods for spherical shallow water equations. *J. Comput. Phys.*, **181**, 499-525.
6. Higueras, I., 2004: On strong stability preserving time discretization methods. *J. Sci. Comput.*, **21**, 193-223.
7. Jorgensen, B. H. , 2003: Tensor formulations of the model equations in strong conservation form for an incompressible flow in general coordinates Technical Report Riso-R-1445, Riso National Laboratory, Roskilde, Denmark.
8. Nair, R. D., S. J. Thomas, and R. D. Loft, 2004: A discontinuous Galerkin global shallow water model. *Mon. Wea. Rev.*, to appear.
9. Prusa, J. M. , and P. K. Smolarkiewicz, 2003: An all-scale anelastic model for geophysical flows: Dynamic grid deformation. *J. Comp. Phys.*, **190**, 601-622.
10. Rancic, M., R. J. Purser, and F. Mesinger, 1996: A global shallow-water model using an expanded spherical cube: Gnomic versus conformal coordinates. *Q. J. R. Meteorol. Soc.*, **122**, 959–982.
11. Sadourny, R., 1972: Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Mon. Wea. Rev.*, **100**, 136–144.
12. Taylor, M., J. Tribbia, M. Iskandarani, 1997: The spectral element method for the shallow water equations on the sphere. *J. Comp. Phys.*, **130**, 92–108.
13. Williamson, D. L., J. B. Drake, J. J. Hack, R. Jakob, P. N. Swarztrauber, 1992: A standard test set for numerical approximations to the shallow water equations in spherical geometry *J. Comp. Phys.*, **102**, 211–224.

Analysis of Discrete Adjoints for Upwind Numerical Schemes

Zheng Liu and Adrian Sandu

Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061
{liuzheng, sandu}@vt.edu

Abstract. This paper discusses several aspects related to the consistency and stability of the discrete adjoints of upwind numerical schemes. First and third order upwind discretizations of the one-dimensional advection equation are considered in both the finite difference and finite volume formulations. We show that the discrete adjoints may lose consistency and stability near the points where upwinding is changed, and near inflow boundaries where the numerical scheme is changed. The impact of adjoint inconsistency and instability on data assimilation is analyzed.

1 Introduction

Adjoint models [3] are widely used in control and in data assimilation in conjunction with the optimization of large scale models. The adjoint methodology can efficiently provide gradients of objective functionals that are formulated in terms of the state of the model.

There are two ways to derive adjoint models [8]. The continuous approach solves numerically the adjoint equation derived from the forward model. The discrete approach formulates directly the adjoint of the forward numerical scheme. The discrete approach is highly attractive since the discrete adjoints, in principle, can be generated automatically [1].

Discrete adjoints of upwind numerical methods pose particular challenges. Symes and Sei [7] pointed out that the consistency of the numerical scheme is not automatically inherited by its discrete adjoint due to the upwind character of the forward scheme. Giles studied the construction and properties of discrete adjoints for hyperbolic systems with shocks [4, 5]. Homescu and Navon [6] discuss the optimal control of flows with discontinuities.

In this paper we study the consistency and stability of discrete adjoints for upwind numerical schemes. The focus is on the advection equation

$$\frac{\partial C}{\partial t} + \frac{\partial (UC)}{\partial x} = 0 \tag{1}$$

and on the corresponding adjoint equation

$$\frac{\partial \lambda}{\partial t} + U \frac{\partial \lambda}{\partial x} = 0 \tag{2}$$

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 829–836, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Advection is the prototype for hyperbolic problems requiring upwinding, and is in itself a fundamental model with many applications. Our discussion includes the case where the wind field has sources and sinks. This situation arises in the directional split solutions of multidimensional advection equation. The consideration of sources and sinks is also important for being able to later extend the results to nonlinear systems.

The paper is organized as follows. The consistency analysis in carried out in Section 2 and the stability analysis in Section 3. Numerical examples are provided in Section 4 and Section 5 summarizes the main findings of this work.

2 Consistency Analysis

In this section we consider several upwind discretizations of (1) and study the consistency of their discrete adjoint schemes with the continuous equation (2).

2.1 First Order Finite Difference Scheme

Forward Scheme. We start with the first order upwind discretization

$$\begin{gathered} C_i' = (1/\Delta x)\ \left[\gamma_i^+ f_{i-1} + (\gamma_i^- - \gamma_i^+) f_i - \gamma_i^- f_{i+1}\right]\ , \\ \gamma_i^+ = 1 \text{ if } U_i \geq 0\ , \quad \gamma_i^+ = 0 \text{ if } U_i < 0\ , \quad \gamma^- = 1 - \gamma^+\ , \quad f_i = U_i C_i\ . \end{gathered} \tag{3}$$

The Dirichlet boundary conditions are $C_0 = C_{LB}$ and $C_{N+1} = C_{RB}$. Here, and throughout the paper, C' denotes the time derivative of C.

Adjoint Scheme. The corresponding discrete adjoint is

$$\lambda_i' = (U_i/\Delta x)\ \left[-\gamma_{i-1}^- \lambda_{i-1} + (\gamma_i^- - \gamma_i^+)\lambda_i + \gamma_{i+1}^+ \lambda_{i+1}\right] \tag{4}$$

with the boundary conditions $\lambda_0 = \lambda_{N+1} = 0$.

Consistency inside the domain. If the wind direction is the same throughout the stencil *(i-1,i,i+1)* then equation (4) is simply the first order spatial discretization method applied to (2). In this case consistency is assured.

We now consider the case where a shift in the wind direction occurs inside the stencil, e.g., $U_{i-1} < 0$ and $U_i, U_{i+1} \geq 0$. The discrete adjoint (4) is:

$$\lambda_i' = (U_i/\Delta x)\ \left[-\lambda_{i-1} - \lambda_i + \lambda_{i+1}\right] \tag{5}$$

The Taylor expansion of the above scheme around the i^{th} node reveals that the the method is formally inconsistent with the continuous adjoint equation. The continuous wind velocity is small around the point of sign change, $U = \mathcal{O}(\Delta x^m)$, and from (5) we have that $\lambda' = \mathcal{O}(\Delta x^{m-1})$. The consistency is maintained if $m > 1$, and is lost for $m \leq 1$. The latter situation includes the most common situation were $U = \mathcal{O}(\Delta x)$, and also the case of a discontinuous wind field $U = \mathcal{O}(1)$.

Consistency near the boundary. The adjoint inflow boundary condition for $U_0 < 0$ is $\lambda_0 = 0$. The discrete adjoint scheme near the boundary

$$\lambda_1' = (U_1/\Delta x)\left[-\gamma_0^- \lambda_0 + (\gamma_1^- - \gamma_1^+)\lambda_1 + \gamma_2^+ \lambda_2\right] \tag{6}$$

is consistent with the adjoint equation.

In conclusion the discrete adjoint of the first order upwind scheme is inconsistent near sources and sinks, and is consistent near the boundaries.

2.2 Third Order Finite Difference Scheme

Forward Scheme. We consider the third order biased upwind scheme:

$$\begin{aligned}
C_i' &= (1/\Delta x)\left[-\tfrac{1}{6}\gamma_i^+ f_{i-2} + \left(\tfrac{1}{3} + \tfrac{2}{3}\gamma_i^+\right) f_{i-1} + \tfrac{1}{2}\left(1 - 2\gamma_i^+\right) f_i \right.\\
&\quad \left. + \left(\tfrac{2}{3}\gamma_i^+ - 1\right) f_{i+1} + \tfrac{1}{6}\left(1 - \gamma_i^+\right) f_{i+2}\right] \quad \text{for} \quad i \geq 3\ ,\\
C_1' &= (1/\Delta x)\left[\gamma_1^+ f_{LB} + \left(\tfrac{3}{2} - \tfrac{5}{2}\gamma_1^+\right) f_1 - 2\left(1 - \gamma_1^+\right) f_2 + \tfrac{1}{2}\left(1 - \gamma_1^+\right) f_3\right]\ , \qquad (7)\\
C_2' &= (1/\Delta x)\left[-\tfrac{1}{6}\gamma_2^+ f_{LB} + \left(\tfrac{1}{3} + \tfrac{2}{3}\gamma_2^+\right) f_1 \right.\\
&\quad \left. + \left(\tfrac{1}{2} - \gamma_2^+\right) f_2 + \left(\tfrac{2}{3}\gamma_2^+ - 1\right) f_3 + \tfrac{1}{6}\left(1 - \gamma_2^+\right) f_4\right]\ .
\end{aligned}$$

The formulation corresponds to an inflow left boundary.

Adjoint Scheme. The corresponding discrete adjoint reads:

$$\begin{aligned}
\lambda_i' = (U_i/\Delta x)\left[\tfrac{1}{6}\left(1 - \gamma_{i-2}^+\right)\lambda_{i-2} + \left(\tfrac{2}{3}\gamma_{i-1}^+ - 1\right)\lambda_{i-1}\right.\\
\left. + \left(\tfrac{1}{2} - \gamma_i^+\right)\lambda_i + \left(\tfrac{1}{3} + \tfrac{2}{3}\gamma_{i+1}^+\right)\lambda_{i+1} - \tfrac{1}{6}\gamma_{i+2}^+\lambda_{i+2}\right]
\end{aligned} \tag{8}$$

Consistency inside the domain. The analysis is similar to the first order case. If the wind direction is the same throughout the stencil *(i-2,i-1,i,i+1,i+2)* the equation (8) is consistent with the continuous adjoint. If the wind sign changes (i.e., there is a source or a sink) inside the stencil then $U = \mathcal{O}(\Delta x^m)$, and the scheme is inconsistent for $m \leq 1$.

Consistency near the boundary. The numerical scheme (7) shifts to a lower order, smaller stencil method near the boundaries. With an inflow left boundary the discrete adjoint equation for the first node is:

$$\lambda_1' = (U_1/\Delta x)\left[-\lambda_1 + \lambda_2 - \tfrac{1}{6}\lambda_3\right] \tag{9}$$

This should reflect an outflow condition for the adjoint. However (9) is inconsistent with the continuous equation (2).

2.3 Finite Volume Approach

We have also studied the consistency of adjoints based on forward finite volume schemes of the form

$$C_i' = \frac{F_{i-\frac{1}{2}} - F_{i+\frac{1}{2}}}{\Delta x} \ . \tag{10}$$

Specifically, we considered first order fluxes for a staggered grid discretization

$$F_{i-\frac{1}{2}} = U^+_{i-\frac{1}{2}} C_{i-1} + U^-_{i-\frac{1}{2}} C_i \ , \tag{11}$$

as well as third order fluxes

$$\begin{aligned} F_{i-\frac{1}{2}} &= -\tfrac{1}{6} U^+_{i-\frac{1}{2}} C_{i-2} + \left(\tfrac{5}{6} U^+_{i-\frac{1}{2}} + \tfrac{1}{3} U^-_{i-\frac{1}{2}} \right) C_{i-1} \\ &\quad + \left(\tfrac{1}{3} U^+_{i-\frac{1}{2}} + \tfrac{5}{6} U^-_{i-\frac{1}{2}} \right) C_i - \tfrac{1}{6} U^-_{i-\frac{1}{2}} C_{i+1} \quad \text{for} \ \ i \geq 3 \\ F_{\frac{1}{2}} &= U^+_{\frac{1}{2}} C_{LB} + \tfrac{3}{2} U^-_{\frac{1}{2}} C_1 - \tfrac{1}{2} U^-_{\frac{1}{2}} C_2 \\ F_{\frac{3}{2}} &= \left(\tfrac{1}{2} U^+_{\frac{3}{2}} + \tfrac{1}{3} U^-_{\frac{3}{2}} \right) C_1 + \left(\tfrac{1}{2} U^+_{\frac{3}{2}} + \tfrac{5}{6} U^-_{\frac{3}{2}} \right) C_2 - \tfrac{1}{6} U^-_{\frac{3}{2}} C_3 \end{aligned} \tag{12}$$

A similar analysis revealed that: (1) the discrete adjoint of the first order finite volume scheme is consistent with the adjoint equation except for the points where there is a sink or source; (2) the discrete adjoint of the first order finite volume scheme is consistent at the boundaries; (3) the discrete adjoint of the third order finite volume scheme is consistent except for the case when there are sinks or sources of velocity within the stencil (when it becomes inconsistent); and (4)the discrete adjoint of the third order finite volume scheme is not consistent at nodes near the inflow boundary.

3 Stability Analysis

In this section the von Neumann stability analysis of the discrete adjoints is carried out for the finite difference formulations.

The first order spatial discretization leads to the system (3) of ordinary differential equations, which is resolved in time using the forward Euler method. The Courant-Friedrichs-Levy (CFL) stability condition for the fully discrete forward scheme is $\sigma \leq 1$, where σ is the Courant number. The adjoint of the fully discrete scheme is the equation (4) resolved in time with the forward Euler method.

The system of ordinary differential equations resulting from the third order spatial discretization (7) is resolved in time with the strongly stable, two stage, second order explicit Runge-Kutta method

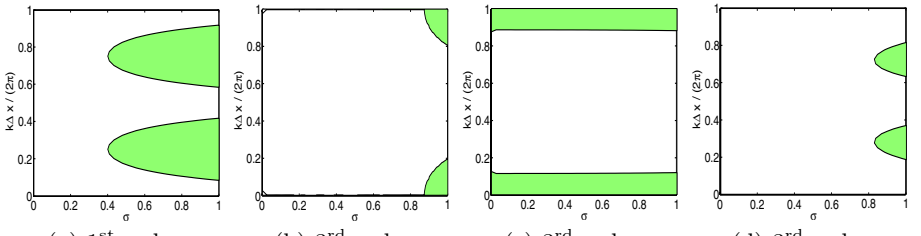
$$C^\alpha = C^n + A\, C^n \ , \qquad C^\beta = C^\alpha + A\, C^\alpha \ , \qquad C^{n+1} = \tfrac{1}{2}(C^n + C^\beta) \ . \tag{13}$$

The CFL stability condition for the fully discrete forward scheme is $\sigma \leq 0.87$. The corresponding discrete adjoint is the equation (8), resolved in time with the same Runge-Kutta scheme (13).

We are concerned with the stability of the resulting discrete adjoints. In the interest of brevity we omit the derivation and present directly the results. The von Neumann stability analysis reveals the dependency of the amplification factors on the Courant number (σ) and on the wave number ($\theta = k\Delta x/(2\pi)$). Fig. 1 presents the areas of stability (white) and instability (gray) in the (σ, θ) plane for different situations.

The stability of the first order discrete adjoint is the same as for the forward method when all wind directions in the stencil *(i-1,i,i+1)* are the same. However, as seen in Fig. 1(a), when the wind changes direction inside the stencil a stricter stability condition ($\sigma \leq 0.4$) is required.

Fig. 1(b) reveals that the stability of the third order discrete adjoint is the same as for the forward method when the wind does not change direction within the stencil *(i-2,$\cdots$,i+2)*. Fig. 1(d) shows that the discrete adjoint is stable with a slightly restricted Courant number if the wind is negative in the leftmost two stencil points. An interesting situation arises when the wind is negative in only the leftmost grid point. In this case the discrete adjoint is unconditionally unstable as seen in Fig. 1(c).



(a) 1st order $U_{i-1} < 0$ $U_i, U_{i+1} \geq 0$ (b) 3rd order $U_{i-2} \cdots U_{i+2} \geq 0$ (c) 3rd order $U_{i-2} < 0$; $U_{i-1} \cdots U_{i+2} \geq 0$ (d) 3rd order $U_{i-2}, U_{i-1} < 0$; $U_i \cdots U_{i+2} \geq 0$

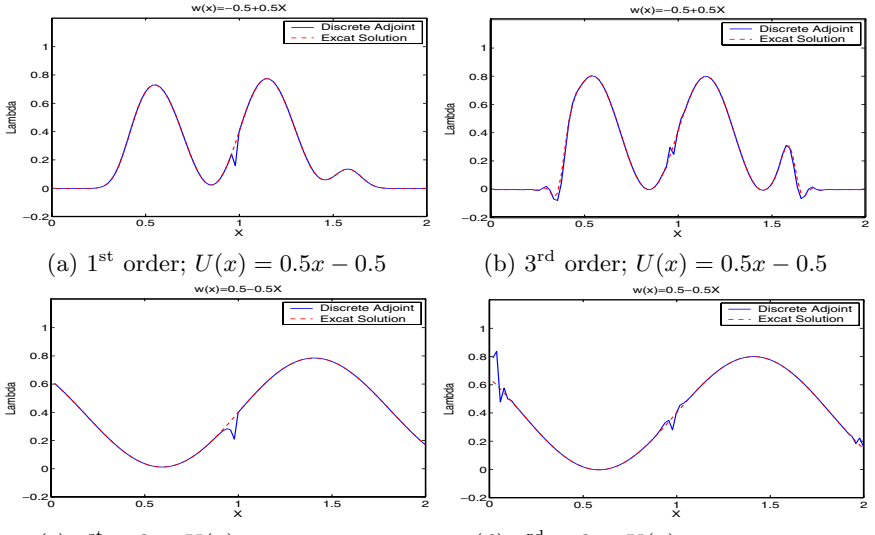
Fig. 1. The stable (white) and unstable (gray) regions of the discrete adjoints in the Courant number (σ) – wave number ($\theta = k\Delta x/(2\pi)$) plane

4 Numerical Examples

To illustrate the inconsistency we use the discrete adjoint method to solve the adjoint equation. The initial condition is a sine function, and the numerical adjoint solutions after backward integration are shown in Fig. 2.

First we consider a linear wind field that has negative values in left half and positive values in the right half of the domain. The center is a source with zero velocity and both boundaries are outflow boundaries. Both the first order (Fig. 2(a)) and third order discrete adjoint solutions (Fig. 2(b)) are corrupted by the inconsistency near the central source.

Next we consider a linear wind field with opposite sign. The center of the domain is now a sink with zero velocity and both boundaries are inflow boundaries. Both the first order (Fig. 2(c)) and third order discrete adjoint solutions (Fig. 2(d)) are corrupted by the inconsistency near the central sink. Moreover, the third order discrete adjoint is inconsistent near the boundaries (Fig. 2(d)) .



(a) 1st order; $U(x) = 0.5x - 0.5$ (b) 3rd order; $U(x) = 0.5x - 0.5$

(c) 1st order; $U(x) = -0.5x + 0.5$ (d) 3rd order; $U(x) = -0.5x + 0.5$

Fig. 2. The discrete adjoint solutions is not consistent with the continuous adjoint near sources/sinks. In addition, the third order solution is also inconsistent near inflow boundaries

We now assess the influence of inconsistency and instability on variational data assimilation. The limited-memory quasi-Newton optimization algorithm L-BFGS [2] is used to recover the initial conditions from observations of the solution at selected grid points and time moments. Assimilation experiments (not reported here) revealed that the convergence rates of the optimization process are nearly identical with discrete and with continuous adjoints as long as the Courant number is small. Consequently the inconsistency and instability of the discrete adjoints do not seem to impact visibly the data assimilation process.

The results of several assimilation experiments are shown in Fig. 3. Fig. 3(a) corresponds to a constant, positive wind field. The inflow (left) boundary adjoint inconsistency does not affect the optimization and the recovered initial condition matches well the reference. Fig. 3(b) is for a piecewise constant wind field, with a source in the center and two outflow boundaries. The recovery near the source is good, however the recovery near the boundaries is poor. This is due to the information loss when the "particles" exit the domain. Fig. 3(c) considers a linear wind field, with a source of zero velocity in the center and two outflow

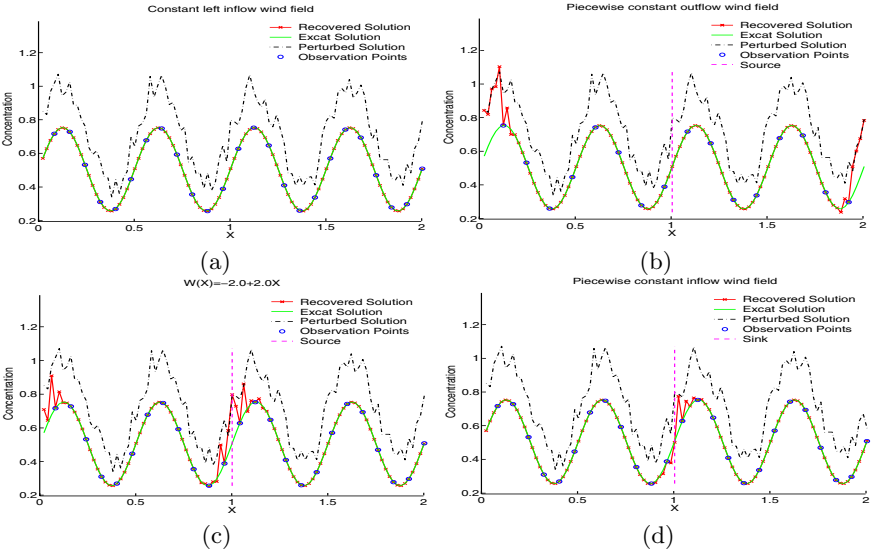


Fig. 3. (a) Constant, positive wind field; (b) Piecewise constant wind field, with a source in the center and two outflow boundaries; (c) Linear wind field, with a source of zero velocity in the center and two outflow boundaries; (d) Piecewise constant wind field, with a sink in the center and two inflow boundaries

boundaries. The recovery of the left outflow boundary is inaccurate, as expected. The recovery of the right outflow boundary is accurate due to the observation point placed on the boundary grid point. The error in recovered initial condition near the center is due to the very small wind speed. The "particles" cannot reach the nearest observation point during the simulation, which results in information loss. Fig. 3(d) is for piecewise constant wind field, with a sink in the center and two inflow boundaries. The recovery error near the center is due to the information loss when the "particles" disappear into the sink (note that there is no observation point at the sink). Both sinks and sources lead to inconsistent discrete adjoints, however the recovery of the initial condition is difficult only for the sinks. Consequently, the recovery error is due to information loss and not to inconsistency.

5 Conclusions

In this paper we analyze the consistency and stability of discrete adjoints for upwind numerical schemes. The focus is on first and third order upwind discretizations of the one-dimensional advection equation.

The discrete adjoints are inconsistent with the continuous adjoint equation at inflow boundaries and near sinks or sources (i.e., points where the wind field changes sign). The von Neumann stability of the forward numerical scheme is

not automatically maintained by the adjoint operation. Depending on the upwinding direction of different points inside the stencil the discrete adjoint can be: (1) linearly stable with a CFL time step restriction similar to that of the forward method, (2) linearly stable under a more stringent CFL condition, or (3) unconditionally unstable at a given point near sources or sinks.

The inconsistency and instability do not affect the performance of the optimization procedure in the data assimilation examples considered here. Both discrete and continuous adjoints lead to similar convergence rates for the recovery of the initial conditions. However, the optimization process is hindered by the loss of information occurring when: (1) the solution collapses into a sink or a shock; (2) the solution exits the domain through an outflow boundary; and (3) the solution features propagate only on a short distance, insufficient to reach one of the observation sites.

Acknowledgements

This work was supported by the National Science Foundation through the awards NSF CAREER ACI 0093139 and NSF ITR AP&IM 0205198.

References

1. *Evaluating derivatives: Principles and Techniques of Algorithmic Differentiation*, volume 41 of *Frontiers in Applied Mathematics*. SIAM, 2000.
2. R. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal of Scientific and Statistical Computing*, 16(5):1190–1208, 1995.
3. D. G. Cacuci. Sensitivity theory for nonlinear systems. I. nonlinear functional analysis approach. *Journal of Mathematical Physics*, 22:2794–2802, 1981.
4. M.B. Giles. Discrete adjoint approximations with shocks. Technical Report 02/10, Oxford University Computing Laboratory, Numerical Analysis Group, 2002.
5. M.B. Giles, M.C. Duta, and N.A. Pierce. Algorithm developments for discrete adjoint methods. *AIAA Journal*, 41(2):198–205, Feburary 2003.
6. C. Homescu and I.M. Navon. Optimal control of flow with discontinuities. *Journal of Computational Physics*, 187:660–682, 2003.
7. A. Sei and Symes W. A note on consistency and adjointness of numerical schemes. Technical Report CSRPC-TR95527, Center for Research in Parallel Computation, Rice University, January 1995.
8. Z. Sirkes and E. Tziperman. Finite difference of adjoint or adjoint of finite difference? *Monthly Weather Review*, 49:5–40, 1997.

The Impact of Background Error on Incomplete Observations for 4D-Var Data Assimilation with the FSU GSM

I. Michael Navon[1], Dacian N. Daescu[2], and Zhuo Liu[1]

[1] School of Computational Science and Information Technology, Florida State University, Tallahassee, FL
navon@csit.fsu.edu
[2] Dept. of Mathematics and Statistics, Portland State University, Portland, OR

Abstract. To assess the impact of incomplete observations on the 4D-Var data assimilation, twin experiments were carried out with the dynamical core of the new FSU GSM consisting of a T126L14 global spectral model in a MPI parallel environment. Results and qualitative aspects are presented for incomplete data in the spatial dimension and for incomplete data in time, with and without inclusion of the background term into the cost functional. The importance of the background estimate on the 4D-Var analysis in the presence of small Gaussian errors in incomplete data is also investigated.

Keywords: Data assimilation, incomplete observations, background error.

1 Introduction

A major issue in data assimilation is that the observing system providing full coverage, i.e., satellites rely on tuning procedures based on the radiosonde observing network and therefore are not well tuned over regions where the radiosonde network is sparse. In the southern hemisphere and tropics where most of the surface is covered by oceans, observations are sparse lacking density and uniformity of conventional wind-profile data coverage distribution available in Northern hemisphere.

In this paper, a new MPI-based parallel version of FSU global spectral model and its adjoint is used to study the impact of incomplete observations in space and time dimension on variational data assimilation. The impact of inclusion of a background error covariance term in the 4-D Var data assimilation cost functional formulation is analyzed. It is crucial for the performance of the analysis system to include the background term to ensure that observations produce statistically consistent increments for model variables that are both smooth and balanced. When data sparse areas or data void areas are present, the background propagates information from observations at earlier times into the data voids.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 837–844, 2005.
© Springer-Verlag Berlin Heidelberg 2005

This means that correlations in background error covariance matrix **B** will perform spatial spreading of information from observation points to a finite domain surrounding them.

Measurement of background error covariances has been addressed in the work of Hollingsworth and Lonnberg [2], Parrish and Derber [7], and Ingleby [3], to name a few. Since the background error covariance matrix is of huge dimensions, efficient ways to estimate it need to be derived. The inverse covariance matrix of **B** is represented via control variable transforms in order to obtain a simplification. See for instance Lorenc et al. [6].

The structure of this paper is as follows. In section 2, we briefly present a basic description of the FSU Global Spectral Model and its implementation. The formulation of the 4D variational data assimilation problem and the specification of the background error covariance matrix are discussed in section 3. In section 4 numerical results with incomplete observations in both spatial and temporal domains are described in a twin experiments setup. Assimilation of sparse noisy data is also discussed. Summary and conclusions are presented in section 5.

2 Short Description of the FSU GSM

The FSU GSM is a spectral global hydrostatic primitive equation model. The prognostic variables are vorticity, divergence, virtual temperature, moisture and logarithm of surface pressure. The model uses the spectral technique in the horizontal direction, and second order finite difference in the vertical. For details we refer to Krishnamurti *et al.* [4]. The wave number truncation used in the numerical experiments is T126 for real time forecasts. Higher resolutions may be used for research purposes. A σ coordinate system with 14 σ-levels is used in the vertical. The model physics include long and shortwave radiation, boundary layer processes, large scale precipitation and shallow and deep cumulus convection.

3 The Background Error in 4D-Var Data Assimilation

One of the important issues in variational data assimilation is the specification of the background error covariance matrix. In recent years, a number of research efforts were dedicated to the study of what is known as the background error term denoted by J_b and usually included in the definition of the basic cost function

$$J_b(\mathbf{X_0}) = \frac{1}{2}(\mathbf{X_0} - \mathbf{X}_b)^T \mathbf{B}^{-1}(\mathbf{X_0} - \mathbf{X}_b) \ .$$

In the equation above $\mathbf{X_0} - \mathbf{X_b}$ represents the departures of the model variables at start of the analysis from the background field $\mathbf{X_b}$; **B** is an approximation to the covariance matrix of background error.

In this case, the cost function J assumes the form

$$J(\mathbf{X_0}) = J_b(\mathbf{X_0}) + J_o(\mathbf{X_0}) \tag{1}$$

where $J_o(\mathbf{X_0})$ is the distance to observations of a forecast initiated from $\mathbf{X_0}$. With the inclusion of background error term, we can prove the uniqueness of the solution of the minimization process with incomplete observations for the linear case [8]. For the nonlinear case, uniqueness may be guaranteed under restricted conditions only.

3.1 Formulation of the Background Error Covariance

The background error covariance matrix is of rather huge dimension , its typical size being of $10^6 \times 10^6$ elements. This is not tractable either from viewpoint of storage requirements or from that of available statistical information.

To avoid calculating the inverse of $\mathbf{B}$, we decided to proceed with a simple form of $\mathbf{B}^{-1}$ by taking $\mathbf{B} = \mathbf{D}^{\frac{1}{2}}\mathbf{C}\mathbf{D}^{\frac{1}{2}}$, where $\mathbf{C}$ was taken to vary only along the horizontal dimensions and $\mathbf{D}$ is a diagonal matrix. We defined

$$\mathbf{C}^{-1} \approx w_0 I + w_2 \nabla^2 \nabla^2 \tag{2}$$

where w_0 and w_2 are chosen such that the power spectrum is similar to that of the Gaussian correlation function (see Daley [1]):

$$\mathbf{C}_{ij} = e^{-\frac{1}{2}\left(\frac{x_i - x_j}{l}\right)^2} \tag{3}$$

where $x_i - x_j$ is the distance between grid points and l is the correlation length scale.

The variance matrix D was chosen to vary in the vertical such that its inverse diminished in the upper levels (M. Zupanski, personal communication).

$$d^{\frac{1}{2}} = \alpha e^{-\beta((p-p_{ref})/1000.)^2}$$

where $d^{\frac{1}{2}}$ represents the square root of the diagonal element of D, which varies only along the vertical coordinate, p is the pressure and p_{ref} is reference pressure which taken the values from 100hPa to 250hPa according to different variables, so the the inverse of the variance will diminish around the upper reference pressure level. α and β are used to adjust the distribution of $d^{\frac{1}{2}}$ along vertical coordinate.

4 Numerical Experiments with Incomplete Observations

Twin experiments were carried out using the dynamical core of the FSU GSM with complete observations which served as the control run. The length of data assimilation window was set to 6 hours from 06UTC Oct. 2, 2002 to 12UTC Oct. 2, 2002.

The observation data were generated by integrating the forward model 6 hours from an initialized ECMWF analysis at 06UTC Oct. 2, 2002 (thereafter referred to as unperturbed initial data), and were available at every timestep and at each Gaussian grid point. The initial guess of the initial condition was

taken from the initialized ECMWF analysis at 00UTC Oct. 2, 2002 (thereafter refer to perturbed initial data), which is 6 hours prior to the initial time level.

A suitable choice of the background state $\mathbf{X}_b$ is important, since it will strongly affect retrieved initial data. In our experiments, the data were generated by integrating the forward model 6 hours prior to the initial time 00UTC Oct. 2, 2002, arriving at same initial time as the start time of the data assimilation window (06UTC Oct. 2, 2002). The limited-memory quasi-Newton method of Liu and Nocedal [5] (L-BFGS) was used to carry out the unconstrained minimization process to obtain the optimal initial data. To simulate the case of incomplete observations, we reduced the number of observations available in the space dimension to every 2, 4 or 8 grid points, respectively. In order to investigate impact of incomplete observations over data void areas, we carried out an experiment where observations over all grid points located over oceans of the Southern hemisphere were missing. In another experiment we reduced the observations in the time dimension to be available only every 2, 4 or 8 timesteps, respectively. The impact of the background term in the retrieval of initial conditions is further analyzed in experiments where small random, Gaussian, noise is added to the incomplete observational data.

4.1 4D-Var Assimilation Experiments Without Background Term

Figure 1 provides the rms errors of height field at 500 hPa, calculated between the fields of model generated observations and the fields obtained by integrating the optimized initial data with incomplete observation after 40 function evaluations (left) and after 100 function evaluations (right). It shows that for incomplete observations in space dimension, the error reduction obtained by the minimization process for incomplete observations depends on density of observations in space dimension.

For observations available every 2 grid points, although the decrease rate of the cost function is slowed down, it can still retrieve the initial data to a certain degree, while for observations available only at every 4 or 8 grid points, the errors increase to a higher degree. The sparser the density of the observations, the larger the departure from observations obtained with minimization of incomplete observations. This issue becomes evident for the data-void area experiment in which case the retrieved initial data were erroneous (i.e. rms difference was quite large). For incomplete observations in the time dimension there were only minor differences in the retrieved initial data for the different cases considered.

4.2 Impact of the Background Error Term on 4D-Var

Since the background error term provides information related to the initial data, we carried out a number of experiments with incomplete observations in the space dimension with a background error term J_b included in the cost function. The results show that the behavior of the minimization process when background covariance term is included in the cost functional is quite different from the one without its inclusion. Figure 2 (left) shows the rms errors of height field at 500

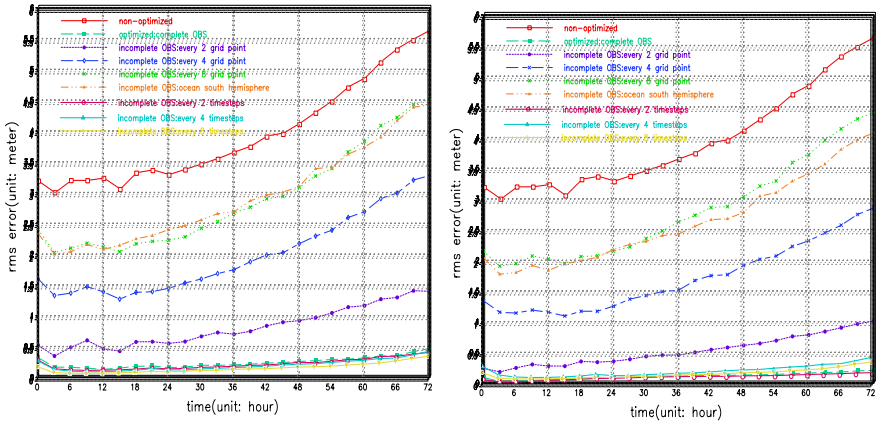


Fig. 1. Time evolution of rms of the height field at 500 hPa after 40 function evaluations (left) and after 100 function evaluations (right) for different cases of incomplete observations without inclusion of background error term, red(□)–non-optimized, *i.e.* result obtained by integrating perturbed initial condition, aqua(■, long dash)–optimized by complete observations, dark purple(•, short dash)–optimized by incomplete observations available every 2 grid points, dark blue(◊, long dash short dash)–optimized by incomplete observations available every 4 grid points, light green(x, dotted)–optimized by incomplete observations available every 8 grid points, orange(▲, dot dot dash)–optimized by incomplete observations that data missing over south hemisphere oceans, magenta(○, solid)–optimized by incomplete observations available every 2 timesteps, light blue(△, solid)–optimized by incomplete observations available every 4 timesteps, yellow(x, solid)–optimized by incomplete observations available every 8 timesteps

hPa, calculated between the fields obtained by integrating the optimized initial data with complete observations and the fields obtained by integrating optimized initial data with incomplete observations after 40 function evaluations. It shows that for incomplete observations in the space dimension, like for the case of exclusion of background error term, the error reduction obtained by the minimization process with incomplete observations depends on the density of the observations in space dimension. We observe that the rms curves after 40 function evaluations exhibit 6- hour oscillations for both incomplete observations in space (observations available every 2-gridpoints) as well as for incomplete observations in time. To assess whether this oscillation is due to the fact that inclusion of background error term altered rate of convergence of the minimization, we carried out an experiment where the rms curves were calculated after 100 function evaluations. The results are also shown in Fig. 2 (right) and we noticed that in this case the abovementioned oscillations vanished which points strongly to the fact that inclusion of background error term has altered structure of spectrum of Hessian of cost functional, thus changing convergence rate characteristics of the minimization.

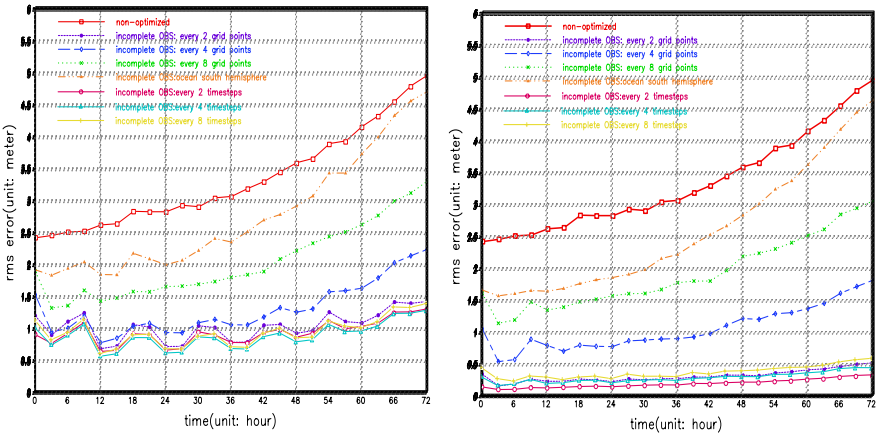


Fig. 2. Time evolution of rms of the height field at 500 hPa after 40 function evaluations (left) and after 100 function evaluation (right) for different cases of incomplete observations with inclusion of background error term, red(□)–non-optimized, *i.e.* result obtained by integrating the perturbed initial condition, dark purple(•, short dash)–optimized by incomplete observations available every 2 grid points, dark blue(◊, long dash short dash)–optimized by incomplete observations available every 4 grid points, light green(x, dotted)–optimized by incomplete observations available every 8 grid points, orange(▲, dot dot dash)–optimized by incomplete observations where data is missing over south hemisphere oceans, magenta(○, solid)–optimized by incomplete observations available every 2 timesteps, light blue(△, solid)–optimized by incomplete observations available every 4 timesteps, yellow(x, solid)–optimized by incomplete observations available every 8 timesteps

4.3 The Impact of the Background Term in the Presence of Errors in Incomplete Observations

In this section numerical experiments are used to investigate the impact of the background term on the retrieval of the initial conditions when a small Gaussian noise of up to 0.1% is added to the incomplete observations. For briefness, we discuss only two representative cases: first, perturbed observations are incomplete in space, available on a sparse subgrid at every 8 grid points for each horizontal level; second, perturbed observations are incomplete in time, available every 8 time steps (each hour since the integration time step is $\Delta t = 450s$). For each case two data assimilation experiments are set up: one without the background term and one with the background term included in the cost functional. The ECMWF analysis at 06UTC Oct. 2, 2002 is used as reference initial state in the qualitative analysis of the results. For computational considerations, we restricted the minimization process to 25 iterations or 30 function evaluations, whichever limit is reached first.

Incomplete Observations in Space. In a first experiment, we consider the assimilation of noisy data, sparse in the spatial dimension, in the absence of the background term. The errors in the retrieved initial conditions for the 500hPa height field ranged from −8m to 10m and we noticed that the analysis errors were distributed throughout the spatial domain with no particular structure and with magnitude about twice as large as the errors in observations. When the background term was included into the cost functional we noticed that the distance to observations increases during the first 24 iterations and the analysis was significantly closer to the background. Therefore we expect a slow assimilation process that may benefit from a better scaling between J_b and J_o. The errors in the retrieved initial conditions for the 500hPa height field typically ranged from −10m to 10m.

Incomplete Observations in Time. Assimilation of the noisy data, incomplete in time, and without background term provided improved results as compared to the noisy data, incomplete in space experiment. In this case, the errors in the retrieved initial conditions for the 500hPa height field ranged from −5m to 4m. The experiment with the background term included into the cost functional provided an analysis closer to both background term and observations. Errors in the retrieved initial conditions for the 500hPa height field typically ranged from −10m to 10m.

5 Summary and Conclusions

We analyzed the impact of inclusion of the background error term for incomplete observations in either space or time in the framework of 4-D Var data assimilation with the FSU GSM and its adjoint model. First we carried out the experiments on the impact of incomplete observations in absence of the background error term for 4-D Var data assimilation with the FSU GSM. Results show that for incomplete observations in space dimension, the minimization process fails to successfully retrieve the initial data, while for incomplete observations in time dimension, the minimization process can retrieve the initial data. Then we carried out a series of experiments on the impact of background term on incomplete observations for 4-D Var data assimilation with the FSU GSM. For the sake of simplification, we calculated the inverse of the background covariance matrix $\mathbf{B}^{-1}$ directly by using a diffusion operator. This avoided the calculation of the inverse of a huge dimension matrix.

Results obtained show that, inclusion of the background error term had a positive impact on convergence of minimization for incomplete observations in space dimension. The sparser the incomplete observations in space dimension, the stronger was the impact of the background error term. However for the case of a data void over the southern hemisphere oceans, the convergence of minimization was observed to be slowed-down.

In contrast to the case of incomplete observations in the space dimension, the background error term had a negative impact on the convergence of minimiza-

tion for incomplete observations in time dimension. The sparser the incomplete observations in time dimension, the larger the negative impact of the background error term. The time evolution of the rms error of the height field at 500 hPa for a 72-hours forecast for different cases of incomplete observations with and without inclusion of the background error term were discussed.

Numerical experiments with small noise added to the incomplete observations were also considered. In the absence of the background term, we noticed that errors in incomplete observations in space resulted in larger errors in the analysis estimate, whereas errors in incomplete observations in time resulted in errors of similar magnitude in the analysis estimate. When the background term was included, assimilation of noisy incomplete data in space resulted into a slow optimization process, with the analysis state close to the background estimate and further from data. By contrast, the analysis state provided by the assimilation of incomplete data in time provided an analysis closer to both background and observations.

These experiments also indicate that in the case of incomplete observations the specification of the background estimate becomes of crucial importance to the analysis. Extension of this study to a full physics version with realistic observations should provide additional insight about the role played by the background error in 4-D Var with incomplete observations.

Acknowledgements

This work was funded by NSF Grant ATM-0201808. The authors would like to thank Dr. Linda Peng the grant manager. We would like to thank Dr. Milija Zupanski for his helpful and insightful advice and his generous sharing of his code.

References

1. Daley, R.: Atmospheric Data Analysis. Cambridge University Press, New York (1991)
2. Hollingsworth, A. and Lönnberg, P.: The statistical structure of short-range forecast errors as determined from radiosonde data. Part I:the wind field. *Tellus* **38A** (1986) 111–136.
3. Ingleby, N.B.: The statistical structure of forecast errors and its representation in the Met Office global 3D variational data assimilation scheme,*Q.J.R. Meteorol. Soc.* **127** (2001) 209–231.
4. Krishnamurti T.N., Bedi, H.S. and Hardiker, V.M.: An introduction to global spectral modeling, Oxford University Press (1998)
5. Liu, D.C., and Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Math. Programm.* **45** (1989) 503–528.
6. Lorenc, A.C. and Coauthors: The Met Office global 3D variational data assimilation scheme. *Q. J. R. Meteorol. Soc.* **126** (2000) 2991–3012.
7. Parrish, D.F. and Derber, J.: The national meteorological center's spectral statistical-interpolation analysis system. *Mon. Wea. Rev.* **120** (1992) 1747–1763.
8. Zou, X., Navon, I.M. and Le-Dimet, F.X.: Incomplete observations and control of gravity waves in variational data assimilation. *Tellus* **44A** (1992) 273–296.

Disjoint Segments with Maximum Density

Yen Hung Chen[1], Hsueh-I Lu[2,⋆] and Chuan Yi Tang[1]

[1] Department of Computer Science,
National Tsing Hua University, Hsinchu 300, Taiwan, R.O.C.
{dr884336, cytang}@cs.nthu.edu.tw
[2] Department of Computer Science and Information Engineering,
National Taiwan University, Taipei 106, Taiwan, R.O.C.
hil@csie.ntu.edu.tw

Abstract. Given a sequence A of numbers and two positive integers ℓ and k, we study the problem to find k disjoint segments of A, each has length at least ℓ, such that their sum of densities is maximized. We give the first known polynomial-time algorithm for the problem: For general k, our algorithm runs in $O(n\ell k)$ time. For the special case with $k = 2$ (respectively, $k = 3$), we also show how to solve the problem in $O(n)$ (respectively, $O(n + \ell^2)$) time.

1 Introduction

Let $A = \langle a_1, a_2, \ldots, a_n \rangle$ be the input sequence of n numbers. Let $A_{i,j}$ denote the consecutive subsequence $\langle a_i, a_{i+1}, \ldots, a_j \rangle$ of A. The *length* of $A_{i,j}$, denoted $|A_{i,j}|$, is $j - i + 1$. The *density* of $A_{i,j}$, denoted $d(A_{i,j})$ is $\frac{a_i + a_{i+1} + \cdots + a_j}{j-i+1}$ of $A_{i,j}$. Observe that with an $O(n)$-time preprocessing to compute all $O(n)$ prefix sums $a_1 + a_2 + \cdots + a_j$ of A, the density of any segment $A_{i,j}$ can be obtained in $O(1)$ time.

Two segments $A_{i,j}$ and $A_{i',j'}$ of A are *disjoint* if $i \leq j < i' \leq j'$ or $i' \leq j' < i \leq j$. Two segments of A *overlap* if they are not disjoint. Motivated by the locating GC-rich regions [9, 14, 15, 16], CpG islands [3, 5, 11, 18] in a genomic sequence and annotating multiple sequence alignments [17], Lin, Huang, Jiang and Chao [13] formulated and gave an $O(n \log k)$-time heuristic algorithm for the problem of identifying k disjoint segments of A with maximum sum of densities. Specifically, given two additional positive integers k and ℓ, the problem is to find k disjoint segments of A, each has length at least ℓ, such that the sum of their densities is maximized. We present the first known polynomial-time algorithm to solve the problem. Our algorithm runs in $O(n\ell k)$ time for general k. We also show that the special case with $k = 2$ (respectively, $k = 3$) can be solved in $O(n)$ (respectively, $O(n + \ell^2)$) time.

⋆ The corresponding author. Address: 1 Roosevelt Road, Section 4, Taipei 106, Taiwan, R.O.C. Webpage: www.csie.ntu.edu.tw/∼hil/.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 845–850, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Related work. When $k = 1$, the problem studied in the present paper becomes the extensively studied *maximimum-density segment problem* [2, 6, 9, 10, 12]. The problem for general k is also closely related to the *GTile with bounded number of tiles* problem [1], which is a natural extension of the *maximum-sum segment* problem studied in [12, 4].

The rest of this paper is organized as follows. Section 2 describes our $O(n\ell k)$-time algorithm for general k. Section 3 shows how to solve the case with $k = 2$ in $O(n)$ time. Section 4 shows how to solve the case with $k = 3$ in $O(n + \ell^2)$ time. Section 5 concludes the paper with open questions.

2 Our Algorithm for General k

For a set U of segments, let $D(U) = \sum_{S \in U} d(S)$. A set of segments is *feasible* to our problem if it consists of k disjoint segments of A, each has length at least ℓ. A set U^* of segments is *optimal* if U^* is feasible and $D(U^*) \geq D(U)$ holds for any feasible set U.

Lemma 1. *There exists an optimal set U^* of segments such each segment in U^* has length less than 2ℓ.*

Proof. Suppose that U^* contains a segment $A_{i,j}$ with $|A_{i,j}| \geq 2\ell$. Then, both $U^* \cup \{A_{i,i+\ell-1}\} - \{A_{i,j}\}$ and $U^* \cup \{A_{i+\ell,j}\} - \{A_{i,j}\}$. Moreover, one of them has to be optimal, since $\max(d(A_{i,i+\ell-1}), d(A_{i+\ell,j})) \geq d(A_{i,j})$. We then use the new optimal set to replace the original U^*. The lemma is proved by continuing this process until each segment in the resulting optimal set U^* has length less than 2ℓ.

According to Lemma 1, it suffices to focus on segments with lengths at least ℓ and less than 2ℓ. Let ρ be the number of such segments in A. Clearly, $\rho = O(n\ell)$. Define G to be a graph on these ρ segments such that two nodes in G are adjacent if and only if their corresponding segments overlap in A. Observe that G is an interval graph. Let the weight of each node be the density of its corresponding segment. Then, the problem to compute an optimal set U^* of segments becomes the problem to identify a maximum weight independent set of G that has size k. To the best of our knowledge, no such an algorithm is known, although the version without restriction on the size has been studied in the literature [8, 7].

Our algorithm for identifying an optimal U^* is via the standard technique of dynamic programming as shown below. For each $j = 1, 2, \ldots, n$, let A_j consist of the segments $A_{i,j}$ of A with $1 \leq i \leq j \leq n$ and $\ell \leq |A_{i,j}| < 2\ell$. For each $j = 1, 2, \ldots, n$, let $U^*_{j,t}$ denote a set of t disjoint segments of $A_{1,j}$, each has length at least ℓ and less than 2ℓ, such that $D(U^*_{j,t})$ is maximized. Note that $U^* = U^*_{n,k}$. One can easily compute all $U^*_{j,1}$ with $1 \leq j \leq n$ in $O(n\ell)$ time. For technical reason, if $j < t\ell$, then let $U^*_{j,t} = \emptyset$ and $D(U^*_{j,t}) = -\infty$. To compute all $O(nk)$ entries of $U^*_{j,t}$ in $O(n\ell k)$ time, we use the following straightforward procedure for each $t > 1$ and $j \geq t\ell$.

> Let $U^*_{j,t} = \{A_{s,j}\} \cup U^*_{s-1,t-1}$, where s is an index i that maximizes $d(A_{i,j}) + D(U^*_{i-1,t-1})$ over all indices i such that $A_{i,j}$ is a segment in A_j.

Since each A_j has size $O(\ell)$, if those $U^*_{j,t-1}$ with $j = 1, 2, \ldots, n$ are available, then all $U^*_{j,t}$ with $j = 1, 2, \ldots, n$ can be computed in $O(n\ell)$ time. One can then obtain $U^* = U^*_{n,t}$ in $O(n\ell k)$ time by iterating the above process for $t = 2, 3, \ldots, k$. Therefore, we have the following theorem.

Theorem 1. *It takes $O(n\ell k)$ time to find k disjoint segments of a length-n sequence, each has length at least ℓ, such that the sum of their densities is maximized.*

3 Our Algorithm for $k = 2$

It turns out that the linear time algorithm of Chung and Lu [2] for the case with $k = 1$ can be a useful subroutine to solve the case with $k = 2$ in linear time. For each $i = 1, 2, \ldots, n$, let P_i (respectively, Q_i) be a maximum density segment with length at least ℓ for $A_{1,i}$ (respectively, $A_{i,n}$). Clearly, P_i and Q_{i+1} are disjoint segments of A for each $i = 1, 2, \ldots, n-1$. Chung and Lu's algorithm has the nice feature that can process the input sequence in an online manner. Therefore, all P_i and Q_i with $1 \leq i \leq n$ can be computed by Chung and Lu's algorithm in $O(n)$ time. The set $\{P_i, Q_{i+1}\}$ with maximum $D(\{P_i, Q_{i+1}\})$ is clearly an optimal solution for the case with $k = 2$. Therefore, we have the following theorem.

Theorem 2. *It takes $O(n)$ time to compute a pair of disjoint segments of a length-n sequence, each has length at least ℓ, such that the sum of their densities is maximized.*

4 Our Algorithm for $k = 3$

Suppose that S_{o1}, S_{o2} and S_{o3} form an optimal set of segments for the case with $k = 3$. We first find a maximum-density segment $S_M = A_{m_i,m_j}$ in A. We also compute maximum-density segments $S_L = A_{l_i,l_j}$ in A_{1,m_i-1} and $S_R = A_{r_i,r_j}$ in $A_{m_j+1,n}$, respectively. Then we find the optimal two disjoint density segments $\{S_{L1}, S_{L2}\}$ in A_{1,m_i-1} and $\{S_{R1}, S_{R2}\}$ in $A_{m_j+1,n}$. Let $\{S_{M'}, S_{M''}\}$ be the element in

$$\{\{S_L, S_R\}, \{S_{L1}, S_{L2}\}, \{S_{R1}, S_{R2}\}\}$$

that has maximum sum of densities. Moreover, we find the maximum-density segment $S_{LL} = A_{ll_i,ll_j}$ in A_{1,l_i-1} and the maximum-density segment $S_{RR} = A_{rr_i,rr_j}$ in $A_{r_j+1,n}$. Furthermore, we find the maximum density segment S_{LLL} in A_{1,ll_i-1} and the maximum-density segment S_{RRR} in $A_{rr_j+1,n}$. For brevity, we use $S_x \sim S_y$ (respectively, $S_x \leftrightarrow S_y$) to denote that segments S_x and S_y overlap (respectively, are disjoint). Let U be the set of segments which are intersect to S_M with length from ℓ to $2\ell - 1$. Finally, for each segment S in U, we perform the following Algorithm 1 to find three disjoint segments $\{S_1, S_2, S_3\}$ with $\{S_1, S_2, S_3\} \cap S \neq \emptyset$.

Algorithm 1:

1. For each segment $S_v = A_{v_i,v_j}$ in U, let $S_2 = S_v$. **do**

1.1. (Case 1: $S_v \sim a_{m_i}$ but $S_v \leftrightarrow a_{m_j}$): Find the maximum-density segment $S_{R'}$ in $A_{v_j+1,m_j+2\ell-2}$. Then let $S_3 = S_{R'}$.

If $S_v \leftrightarrow S_L$ **then** $S_1 = S_L$

else

If $S_v \sim S_L$ but $S_v \leftrightarrow S_{LL}$ **then** find the maximum-density segment $S_{L'}$ in $A_{l_i-2\ell+2,v_i-1}$ then let S_1 be the maximum density segment between $S_{L'}$ and S_{LL}.

else find the maximum-density segment $S_{L'}$ in $A_{ll_i-2\ell+2,v_i-1}$ then let S_1 be the maximum density segment between $S_{L'}$ and S_{LLL}.

1.2. (Case 2: $S_v \sim a_{m_j}$ but $S_v \leftrightarrow a_{m_i}$): Find the maximum-density segment $S_{L''}$ in $A_{m_i-2\ell+2,v_i-1}$. Then let $S_1 = S_{L''}$.

If $S_v \leftrightarrow S_R$ **then** let $S_3 = S_R$

else

If $S_v \sim S_R$ but $S_v \leftrightarrow S_{RR}$ **then** find the maximum-density segment $S_{R''}$ in $A_{v_j+1,r_j+2\ell-2}$ then let S_3 be the maximum density segment between $S_{R''}$ and S_{RR}.

else find the maximum-density segment $S_{R''}$ in $A_{v_j+1,rr_j+2\ell-2}$ then let S_3 be the maximum density segment between $S_{R''}$ and S_{RRR}.

1.3. (Case 3: $S_v \subset S_m$): Find the maximum-density segments $S_{L'''}$ and $S_{R'''}$ in $A_{m_i-2\ell+2,v_i-1}$ and $A_{v_j+1,m_j+2\ell-2}$. Let $\{S_1, S_3\} = \{S_{L'''}, S_{R'''}\}$.

end for

2. Let $\{S_a, S_b, S_c\}$ be the maximum total density segments in all these three disjoint segments $\{S_1, S_2, S_3\}$.

Finally, if

$$D(\{S_a, S_b, S_c\}) \leq D(\{S_M, S_{M'}, S_{M''}\}),$$

then let $\{S_{o1}, S_{o2}, S_{o3}\}$ be $\{S_{M'}, S_M, S_{M''}\}$; otherwise, let $\{S_{o1}, S_{o2}, S_{o3}\}$ be $\{S_a, S_b, S_c\}$. Though there are $O(\ell^2)$ iterations in Algorithm 1, we only need $O(\ell^2)$ time in total. We can pre-process to find all $S_{R'}$ in case 1, all $S_{R'''}$ in case 3, all $S_{L''}$ in case 2 and all $S_{L'''}$ in case 3 in $O(\ell^2)$ time. Because the lengths of $A_{m_i-2\ell+2,v_i-1}$ and $A_{v_j+1,m_j+2\ell-2}$ are $O(\ell)$ and the length of S_M is at most $2\ell - 1$. Also pre-process to find all $S_{L'}$ in case 1 and all $S_{R''}$ in case 2 take $O(\ell^2)$ time. As a result, the time complexity of Algorithm 1 is $O(\ell^2)$.

Theorem 3. *It takes $O(n + \ell^2)$ time to compute three disjoint segments of a length-n sequence, each has length at least ℓ, such that the sum of their densities is maximized.*

Proof. Since the time complexity of Algorithm 1 is $O(\ell^2)$, our algorithm runs in $O(n + \ell^2)$ time. It remains to prove the correctness of our algorithm. For any three disjoint segments $\{S_1, S_2, S_3\}$ in A, we will show

$$D(\{S_{o1}, S_{o2}, S_{o3}\}) \geq D(\{S_1, S_2, S_3\}).$$

For convenience, let S_1 be the left segment, let S_2 be the middle segment, and let S_3 be the right segment for the three disjoint segments $\{S_1, S_2, S_3\}$ in A. First, if each of S_1, S_2 and S_3 does not overlap with S_M, then

$$D(\{S_M, S_{M'}, S_{M''}\}) \geq D(\{S_1, S_2, S_3\}).$$

If only one segment of $\{S_1, S_2, S_3\}$ overlaps with S_M, then

$$D(\{S_M, S_{M'}, S_{M''}\}) \geq D(\{S_1, S_2, S_3\}).$$

Hence, the rest of the proof assumes that at least two segments of $\{S_1, S_2, S_3\}$ overlaps with S_M and

$$D(\{S_1, S_2, S_3\}) > D(\{S_{M'}, S_M, S_{M''}\}).$$

Without loss of generality, we may assume that segment $S_2 = S_v = A_{v_i,v_j}$ overlaps with S_M. Then we consider the following three cases. Case 1: $S_v \sim a_{m_i}$ but $S_v \leftrightarrow a_{m_j}$, case 2: $S_v \sim a_{m_j}$ but $S_v \leftrightarrow a_{m_i}$, and case 3: $S_v \subset S_m$. We prove the result for case 1 and case 3. The case 2 can be shown similar to case 1. For case 1, let $S_{R'}$ is the maximum-density segment in $A_{v_j+1,m_j+2\ell-2}$ and $S_3 = S_{R'}$. Because $d(S_1) \leq d(S_L)$ and $d(S_2) \leq d(S_M)$, the segment S_3 must be a subsequence in $A_{v_j+1,m_j+2\ell-2}$; otherwise, we have

$$D(\{S_L, S_M, S_R\}) \geq D(\{S_1, S_2, S_3\}).$$

Hence, we only choose a best S_1 in A_{1,v_i-1}. We consider the following three cases. (1) if $S_v \leftrightarrow S_L$, we only let $S_1 = S_L$ because S_L is the maximum-density segment in A_{1,m_i-1}. (2) If $S_v \sim S_L$ but $S_v \leftrightarrow S_{LL}$. For S_1, we only consider the segments S_{LL} and $S_{L'}$, where $S_{L'}$ is a maximum-density segment in $A_{l_i-2\ell+2,v_i-1}$. Because $S_1 \sim S_L$, segment S_1 is either in A_{1,l_i-1} or in $A_{l_i-2\ell+2,v_i-1}$. (3) $S_v \sim S_L$ and S_{LL}. For S_1, we only consider the segments S_{LLL} and $S_{L'}$, where $S_{L'}$ is a maximum-density segment in $A_{ll_i-2\ell+2,v_i-1}$. Because $S_1 \sim S_{LL}$, segment S_1 is either in A_{1,ll_i-1} or in $A_{ll_i-2\ell+2,v_i-1}$. For case 3, let $S_{L'''}$ is the maximum-density segment in $A_{m_i-2\ell+2,v_i-1}$ and $S_{R'''}$ is the maximum-density segment in $A_{v_j+1,m_j+2\ell-2}$. Because $d(S_v) \leq d(S_M)$, we only let $\{S_1, S_2, S_3\} = \{S_{L'''}, S_v, S_{R'''}\}$. Otherwise, we have

$$D(\{S_L, S_M, S_R\}) \geq D(\{S_1, S_2, S_3\}).$$

□

5 Conclusion

We have shown the first known polynomial-time algorithm to compute multiple disjoint segments whose sum of densities is maximized. An immediate open question is whether the problem can be solved in $o(n\ell k)$ time. Also, it would be interesting to see our techniques for $k = 2, 3$ to be generalized to the cases with larger k.

References

1. P. Berman, P. Bertone, B. DasGupta, M. Gerstein, M.-Y. Kao and M. Snyder: Fast Optimal Genome Tiling with Applications to Microarray Design and Homology Search. *Journal of Computational Biology*, 11:766–785, 2004.
2. K.-M. Chung and H.-I. Lu: An Optimal Algorithm for the Maximum-Density Segment Problem. *SIAM Journal on Computing*, 34:373–387, 2004.
3. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison: *Biological Sequence Analysis*. Cambridge University Press, 1998.
4. T.-H. Fan, S. Lee, H.-I. Lu, T.-S. Tsou, T.-C. Wang, A. Yao: An Optimal Algorithm for Maximum-Sum Segment and Its Application in Bioinformatics. In *Proceedings of the 8th International Conference on Implementation and Application of Automata*, Lecture Notes in Computer Science 2759, 251–257, Santa Barbara, July 2003, Springer-Verlag.
5. M. Gardiner-Garden, and M. Frommer: CpG Islands in Vertebrate Genomes. *Journal of Molecular Biology*, 196:261–282, 1987.
6. M.H. Goldwasser, M.-Y. Kao, and H.-I. Lu: Linear-Time Algorithms for Computing Maximum-Density Sequence Segments with Bioinformatics Applications. *Journal of Computer and System Sciences*, 70:128–144, 2005.
7. U. I. Gupta, D. T. Lee, and J. Y.-T. Leung: Efficient Algorithms for Interval Graphs and Circular-Arc Graphs, *Networks* 12:459–467, 1982.
8. J. Y. Hsiao, C. Y. Tang, and R. S. Chang: An Efficient Algorithm for Finding a Maximum Weight 2-Independent Set on Interval Graphs, *Information Processing Letters*, 43(5):229-235, 1992.
9. X. Huang: An algorithm for Identifying Regions of a DNA Sequence That Satisfy a Content Requirement. *Computer Applications in the Biosciences*, 10:219–225, 1994.
10. S. K. Kim: Linear-Time Algorithm for Finding a Maximum-Density Segment of a Sequence. *Information Processing Letters*, 86:339-342, 2003.
11. F. Larsen, R. Gundersen, R. Lopez, and H. Prydz: CpG Islands as Gene Marker in the Human Genome. *Genomics*, 13:1095–1107, 1992.
12. Y.-L. Lin, T. Jiang, K.-M. Chao: Efficient Algorithms for Locating the Length-Constrained Heaviest Segments with Applications to Biomolecular Sequence Analysis. *Journal of Computer and System Sciences*, 65:570–586, 2002.
13. Y.-L. Lin, X. Huang, T. Jiang, K.-M. Chao: MAVG: Locating Non-overlapping Maximum Average Segments in a Given Sequence. *Bioinformatics*, 19:151–152, 2003.
14. A. Nekrutenko and W.-H. Li: Assessment of Compositional Heterogeneity within and between Eukaryotic Genomes. *Genome Research*, 10:1986–1995, 2000.
15. P. Rice, I. Longden, and A. Bleasby: EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics*, 16:276–277, 2000.
16. P. H. Sellers: Pattern Recognition in Genetic Sequences by Mismatch Density. *Bulletin of Mathematical Biology*, 46:501–514, 1984.
17. N. Stojanovic, L. Florea, C. Riemer, D. Gumucio, J. Slightom, M. Goodman, W. Miller, and R. Hardison: Comparison of Five Methods for Finding Conserved Sequences in Multiple Alignments of Gene Regulatory Regions. *Nucleic Acids Research*, 27:3899–3910, 1999.
18. D. Takai, and P.A. Jones: Comprehensive Analysis of CpG Islands in Human Chromosomes 21 and 22. *Proceedings of the National Academy of Sciences*, 99:3740–3745, 2002.

What Makes the Arc-Preserving Subsequence Problem Hard?*

Guillaume Blin[1], Guillaume Fertin[1], Romeo Rizzi[2], and Stéphane Vialette[3]

[1] LINA - FRE CNRS 2729 Université de Nantes,
2 rue de la Houssinière BP 92208 44322 Nantes Cedex 3 - France
{blin, fertin}@univ-nantes.fr
[2] Università degli Studi di Trento Facoltà di Scienze - Dipartimento di Informatica e Telecomunicazioni
Via Sommarive, 14 - I38050 Povo - Trento (TN) - Italy
Romeo.Rizzi@unitn.it
[3] LRI - UMR CNRS 8623 Faculté des Sciences d'Orsay, Université Paris-Sud
Bât 490, 91405 Orsay Cedex - France
vialette@lri.fr

Abstract. Given two arc-annotated sequences (S, P) and (T, Q) representing RNA structures, the Arc-Preserving Subsequence (APS) problem asks whether (T, Q) can be obtained from (S, P) by deleting some of its bases (together with their incident arcs, if any). In previous studies [3, 6], this problem has been naturally divided into subproblems reflecting intrinsic complexity of arc structures. We show that APS(Crossing, Plain) is **NP**-complete, thereby answering an open problem [6]. Furthermore, to get more insight into where actual border of APS hardness is, we refine APS classical subproblems in much the same way as in [11] and give a complete categorization among various restrictions of APS problem complexity.

Keywords: RNA structures, Arc-Preserving Subsequence, Computational complexity.

1 Introduction

At a molecular state, the understanding of biological mechanisms is subordinated to RNA functions discovery and study. Indeed, it is established that the conformation of a single-stranded RNA molecule (a linear sequence composed of ribonucleotides A, U, C and G, also called *primary structure*) partly determines the molecule function. This conformation results from the folding process due to local pairings between complementary bases ($A-U$ and $C-G$). The RNA secondary structure is a collection of folding patterns that occur in it.

RNA secondary structure comparison is important in many contexts, such as **(i)** identification of highly conserved structures during evolution which suggest a significant common function for the studied RNA molecules [9], **(ii)** RNA

* This work was partially supported by the French-Italian PAI Galileo project number 08484VH and by the CNRS project ACI Masse de Données "NavGraphe".

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 860–868, 2005.
© Springer-Verlag Berlin Heidelberg 2005

classification of various species (phylogeny)[2], **(iii)** RNA folding prediction by considering a set of already known secondary structures [13].

Structure comparison for RNA has thus become a central computational problem bearing many challenging computer science questions. At a theoretical level, RNA structure is often modelled as an *arc-annotated sequence*, that is a pair (S, P) where S is a sequence of ribonucleotides and P represents hydrogen bonds between pairs of elements of S. Different pattern matching and motif search problems have been investigated in the context of arc-annotated sequences among which we can mention Arc-Preserving Subsequence (APS) problem, Edit Distance problem, Arc-Substructure (AST) problem and Longest Arc-Preserving Subsequence (LAPCS) problem (see for instance [3, 8, 7, 6, 1]). For other related studies concerning algorithmic aspects of (protein) structure comparison using *contact maps*, refer to [5, 10].

In this paper, we focus on APS problem: given two arc-annotated sequences (S, P) and (T, Q), this problem asks whether (T, Q) can be exactly obtained from (S, P) by deleting some of its bases together with their incident arcs, if any. This problem is commonly encountered when one is searching for a given RNA pattern in an RNA database [7]. Moreover, from a theoretical point of view, APS problem can be seen as a restricted version of LAPCS problem, and hence has applications in structural comparison of RNA and protein sequences [3, 5, 12]. APS problem has been extensively studied in the past few years [6, 7, 3]. Of course, different restrictions on arc-annotation alter APS computational complexity, and hence this problem has been naturally divided into subproblems reflecting the complexity of the arc structure of both (S, P) and (T, Q): plain, chain, nested, crossing or unlimited (see Section 2 for details). All of them but one have been classified as to whether they are polynomial time solvable or **NP**-complete. The problem of the existence of a polynomial time algorithm for APS(Crossing,Plain) problem was mentioned in [6] as the last open problem in the context of arc-preserving subsequences. Unfortunately, as we shall prove in Section 4, APS(Crossing,Plain) is **NP**-complete even for restricted special cases.

In analyzing the computational complexity of a problem, we are often trying to define a precise boundary between polynomial and **NP**-complete cases. Therefore, as another step towards establishing the precise complexity landscape of APS problem, we consider that it is of great interest to subdivide existing cases into more precise ones, that is to refine classical complexity levels of APS problem, for determining more precisely what makes the problem hard. For that purpose, we use the framework introduced by Vialette [11] in the context of 2-intervals (a simple abstract structure for modelling RNA secondary structures). As a consequence, the number of complexity levels rises from 4 to 8, and all the entries of this new complexity table need to be filled. Previous known results concerning APS problem, along with our **NP**-completeness proofs, allow us to fill all the entries of this new table, therefore determining what exactly makes the APS problem hard.

The paper is organized as follows. Provided with notations and definitions (Section 2), in Section 3 we introduce and explain new refinements of the complexity levels we are going to study. In Section 4, we show that APS($\{\sqsubset, \between\}, \emptyset$) is **NP**-complete thereby proving that *classical* APS(Crossing, Plain) is **NP**-complete as well. As another refinement to that result, we prove that APS($\{<, \between\}, \emptyset$) is **NP**-complete. Finally, in Section 5, we give new polynomial time solvable algorithms for restricted instances of APS(Crossing, Plain).

2 Preliminaries

An RNA structure is commonly represented as an arc-annotated sequence (S, P) where S is a sequence of ribonucleotides (or bases) and P is a set of arcs connecting pairs of bases in S. Let (S, P) and (T, Q) be two arc-annotated sequences such that $|S| \geq |T|$ (in the following, $n = |S|$ and $m = |T|$). APS problem asks whether (T, Q) can be exactly obtained from (S, P) by deleting some of its bases together with their incident arcs, if any.

Since the general problem is easily seen to be intractable [3], the arc structure must be restricted. Evans [3] proposed four possible restrictions on P (resp. Q) which were largely reused in subsequent literature: (1) there is no base incident to more than one arc, (2) there are no arcs crossing, (3) there is no arc contained in another and (4) there is no arc.

These restrictions are used progressively and inclusively to produce five different levels of allowed arc structure: Unlimited - general problem with no restrictions, Crossing - restriction (1); Nested - restrictions (1) and (2); Chain - restrictions (1), (2) and (3); Plain - restriction (4).

Guo proved in [7] that APS(Crossing, Chain) is **NP**-complete. Guo *et al.* observed in [6] that **NP**-completeness of APS(Crossing, Crossing) and APS(Unlimited, Plain) easily follows from results of Evans [3] concerning LAPCS problem. Furthermore, they gave a $O(nm)$ time algorithm for APS (Nested, Nested). This algorithm can be applied to easier problems such as APS(Nested, α) and APS(Chain, α) with $\alpha \in \{$Chain,Plain$\}$. Finally, Guo *et al.* mentioned in [6] that APS(Chain, Plain) can be solved in $O(n+m)$ time. Observe that Unlimited level has no restrictions, and hence is of limited interest in our study. Consequently, from now on we will not be concerned anymore with that level. Until now, the question of the existence of an exact polynomial algorithm for APS(Crossing, Plain) remained open. We will show in the present paper that APS(Crossing,Plain) is **NP**-complete.

3 Refinement of APS Problem

In this section, we propose a refinement of APS problem. We first state formally our approach and explain why such a refinement is relevant for both theoretical and experimental studies.

3.1 Splitting the Levels.

As we will show soon, APS(Crossing, Plain) is **NP**-complete. That result answers the last open problem concerning APS computational complexity with respect to classical complexity levels. However, we are mainly interested in the elaboration of a precise border between **NP**-complete and polynomially solvable cases. Indeed, both theorists and practitioners might naturally ask for more information concerning APS hard cases in order to get valuable insight into what makes the problem difficult. As a next step towards better understanding what makes APS problem hard, we propose to refine classically models used for classifying arc-annotated sequences. Our refinement consists in splitting those models of arc-annotated sequences into more precise relations between arcs. For example, such a refinement provides a general framework for investigating polynomial time solvable and hard restricted instances of APS(Crossing, Plain), thereby refining in many ways Theorem 1 (see Section 5).

We use three relations first introduced by Vialette [11] in the context of 2-*intervals*. Actually, his definition of 2-intervals could almost apply in this paper (the main difference lies in the fact that 2-intervals are used for representing sets of contiguous arcs). Vialette defined three possible relations between 2-intervals that can be used for arc-annotated sequences as-well. They are the following: for any two arcs $A_1 = (i, j)$ and $A_2 = (k, l)$ in P, we will write $A_1 < A_2$ if $i < j < k < l$ (*precedence* relation), $A_1 \sqsubset A_2$ if $k < i < j < l$ (*nested* relation) and $A_1 \between A_2$ if $i < k < j < l$ (*crossing* relation). Two arcs A_1 and A_2 are τ-comparable for some $\tau \in \{<, \sqsubset, \between\}$ if $A_1 \tau A_2$ or $A_2 \tau A_1$. Let $\mathcal{P}$ be a set of arcs and R be a non-empty subset of $\{<, \sqsubset, \between\}$. The set $\mathcal{P}$ is said to be *R-comparable* if any two distinct arcs of $\mathcal{P}$ are τ-comparable for some $\tau \in R$. An arc-annotated sequence (P, A) is said to be an R-arc-annotated sequence for some non-empty subset R of $\{<, \sqsubset, \between\}$ if A is R-comparable. By abuse of notation, we write $R = \emptyset$ in case $A = \emptyset$. Observe that our model cannot deal with arc-annotated sequences which contain only one arc. However, having only one arc or none can not really affect the problem computational complexity. Just one guess reduces from one case to the other.

As a straightforward illustration of above definitions, classical complexity levels for APS problem can be expressed in terms of combinations of our new relations: Plain is fully described by $R = \emptyset$, Chain by $R = \{<\}$, Nested by $R = \{<, \sqsubset\}$ and Crossing by $R = \{<, \sqsubset, \between\}$. The key point is to observe that our refinement allows us to consider new structures for arc-annotated sequences, namely $R \in \{\{\sqsubset\}, \{\between\}, \{<, \between\}, \{\sqsubset, \between\}\}$, which could not be considered using classical complexity levels. Although other refinements may be possible (in particular well-suited for parameterized complexity analysis), we do believe that such an approach allows a more precise analysis of APS complexity. Of course one might object that some of these subdivisions are unlikely to appear in RNA secondary structures. However, it is of great interest to answer, at least partly, the following question: Where is the precise boundary between polynomial and **NP**-complete cases ? Indeed, such a question is relevant for both theoretical and experimental studies.

3.2 Immediate Results

First, observe that we only have to consider cases of APS(R_1,R_2) where R_1 and R_2 are compatible, *i.e.* $R_2 \subseteq R_1$. Indeed, if this is not the case, we can immediately answer negatively since there exists two arcs in T which satisfy a relation in R_2 which is not in R_1, and hence T simply cannot be obtained from S by deleting bases of S. Those useless cases are simply denoted by hatched areas in Table 1.

Some known results allow us to fill many entries of the new complexity table derived from our refinement. The remainder of this subsection is devoted to detailing these first easy statements. We begin with an easy observation concerning complexity propagation properties of APS problem.

Observation 1 *Let R_1, R_2, R'_1 and R'_2 be four subsets of $\{<, \sqsubset, \between\}$ such that $R'_2 \subseteq R_2 \subseteq R_1$ and $R'_2 \subseteq R'_1 \subseteq R_1$. If* APS($R'_1$, R'_2) *is* **NP***-complete (resp.* APS(R_1, R_2) *is polynomial time solvable) then so is* APS(R_1, R_2) *(resp.* APS(R'_1, R'_2)*).*

On the positive side, Guo *et al.* have shown that APS(Nested, Nested) is solvable in $O(nm)$ time [6]. Another way of stating this is to say that APS($\{<, \sqsubset\}$, $\{<, \sqsubset\}$) is solvable in $O(mn)$ time. That result together with Observation 1 may be summarized by saying that APS(R_1, R_2) for any compatible R_1 and R_2 such that $\between \notin R_1$ and $\between \notin R_2$ is polynomial time solvable.
Conversely, Evans has proved that APS(Crossing,Crossing) is **NP**-complete [3]. A simple reading shows that her proof is concerned with $\{<, \sqsubset, \between\}$-arc-annotated sequences, and hence she actually proved that APS($\{<, \sqsubset, \between\}$, $\{<, \sqsubset, \between\}$) is **NP**-complete. Similarly, in proving that APS(Crossing,Chain) is **NP**-complete [7], Guo actually proved that APS($\{<, \sqsubset, \between\}$, $\{<\}$) is **NP**-complete. Note that according to Observation 1, this latter result implies that APS($\{<, \sqsubset, \between\}$, $\{<, \sqsubset\}$) and APS($\{<, \sqsubset, \between\}$,$\{<, \between\}$) are **NP**-complete. Table 1 surveys known and new results for various types of our refined APS problem. Observe that this paper answers all questions concerning APS problem with respect to both classical and new complexity levels.

Table 1. Complexity results after complexity levels refinement. ////: useless cases. $\star$: results from this paper.

APS								
$R_1 \backslash R_2$	$\{<, \sqsubset, \between\}$	$\{\sqsubset, \between\}$	$\{<, \between\}$	$\{\between\}$	$\{<, \sqsubset\}$	$\{\sqsubset\}$	$\{<\}$	$\emptyset$
$\{<, \sqsubset, \between\}$	**NP**-C [3]	**NP**-C $\star$	**NP**-C [7]	**NP**-C $\star$	**NP**-C [7]	**NP**-C $\star$	**NP**-C [7]	**NP**-C $\star$
$\{\sqsubset, \between\}$		**NP**-C $\star$	////	**NP**-C $\star$	////	**NP**-C $\star$	////	**NP**-C $\star$
$\{<, \between\}$			**NP**-C $\star$	**NP**-C $\star$	////	////	**NP**-C $\star$	**NP**-C $\star$
$\{\between\}$				$O(nm^2)$ $\star$	////	////	////	$O(nm^2)$ $\star$
$\{<, \sqsubset\}$					$O(nm)$ [6]	$O(nm)$ [6]	$O(nm)$ [6]	$O(nm)$ [6]
$\{\sqsubset\}$						$O(nm)$ [6]	////	$O(nm)$ [6]
$\{<\}$							$O(nm)$ [6]	$O(n+m)$ [6]
$\emptyset$								$O(n+m)$ [6]

4 Hardness Results

We show in this section that APS($\{\sqsubset, \between\}, \emptyset$) and APS($\{<, \between\}, \emptyset$) are **NP**-complete thereby proving that APS(Crossing, Plain) is **NP**-complete. That result answers an open problem posed by Gramm, Guo and Niedermeier in [6] which is the last open problem concerning the APS computational complexity with respect to classical complexity levels, *i.e.*, Plain, Chain, Nested and Crossing.

Theorem 1. APS($\{\sqsubset, \between\}, \emptyset$) *is* **NP***-complete.*

We prove Theorem 1 by giving a polynomial time reduction from the well known **NP**-complete 3-Sat problem [4]. Due to space considerations, the entire proof is deferred to the full version of the paper.

It follows immediately from Theorem 1 that APS($\{<, \sqsubset, \between\}, \emptyset$), and hence APS(Crossing, Plain), are **NP**-complete. One might naturally ask for more information concerning hard cases of APS problem in order to get valuable insight into what makes the problem difficult. Another refinement of the hardness of APS(Crossing,Plain) is given by the following theorem.

Theorem 2. APS($\{<, \between\}, \emptyset$) *is* **NP***-complete.*

The proof is also by reduction from 3-Sat. Due to space considerations, the rather technical proof is also deferred to the full version of the paper.

5 Two Polynomial Time Solvable APS Problems

We prove in this section that APS($\{\between\}$, $\emptyset$) and APS($\{\between\}$,$\{\between\}$) are polynomial time solvable. In other words, relation $\between$ alone does not imply **NP**-completeness. We need the following notations. Sequences are the concatenation of zero or more elements from an alphabet. We use the period "." as the concatenation operator, but frequently the two operands are simply put side by side. Let $T = T[1]\,T[2]\ldots T[m]$ be a sequence of length m. For all $1 \leq i \leq j \leq m$, we write $T[i:j]$ to denote $T[i]\,T[i+1]\ldots T[j]$. The *reverse* of T is the sequence $T^R = T[m]\ldots T[2]\,T[1]$. A *factorization* of T is any decomposition $T = x_1\,x_2\ldots x_q$ where $x_1, x_2, \ldots x_q$ are (possibly empty) sequences. Let (T, A) be a $\{\between\}$-arc-annotated sequence and $(i, j) \in A$, $i < j$, be an arc. We call $T[i]$ a *forward base* and $T[j]$ a *backward base*. We will denote by LF_T the position of the last forward base in (T, A) and by FB_T the position of the first backward base in (T, A), *i.e.*, $\mathsf{LF}_T = \max\{i : (i, j) \in A\}$ and $\mathsf{FB}_T = \min\{j : (i, j) \in A\}$. By convention, we let $\mathsf{LF}_T = 0$ and $\mathsf{FB}_T = |T| + 1$ if $A = \emptyset$. Observe that $\mathsf{LF}_T < \mathsf{FB}_T$. We begin by proving a factorization result on $\{\between\}$-arc-annotated sequences.

Lemma 1. *Let S and T be two $\{\between\}$-arc-annotated sequences of length n and m, respectively. If T occurs as an arc preserving subsequence in S, then there exists a factorization (possibly trivial) $T[\mathsf{LF}_T + 1 : \mathsf{FB}_T - 1] = xy$ such that $T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$ occurs as an arc preserving subsequence in $S[1 : \mathsf{FB}_S - 1] \cdot S[\mathsf{FB}_S : n]^R$.*

Proof. Suppose that T occurs as an arc preserving subsequence in S. Since both S and T are $\{\langle\rangle\}$-arc-annotated sequences, then there exist two factorizations $S[1 : \mathsf{LF}_S] = uw$ and $S[\mathsf{FB}_S : n] = zv$ such that: (i) $T[1 : \mathsf{LF}_T]$ occurs in u, (ii) $T[\mathsf{LF}_T + 1 : \mathsf{FB}_T - 1]$ occurs in $w \cdot S[\mathsf{LF}_S + 1 : \mathsf{FB}_S - 1] \cdot z$ and (iii) $T[\mathsf{FB}_T : m]$ occurs in v. Then it follows that there exists a factorization $T[\mathsf{LF}_T + 1 : \mathsf{FB}_T - 1] = xy$ such that x occurs in $w \cdot S[\mathsf{LF}_S + 1 : \mathsf{FB}_S - 1]$ and y occurs in z, and hence $T' = T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$ occurs as an arc preserving subsequence in $S' = S[1 : \mathsf{FB}_S - 1] \cdot S[\mathsf{FB}_S : n]^R$ □

Theorem 3. APS($\{\langle\rangle\}$,$\{\langle\rangle\}$) *is solvable in* $O(nm^2)$ *time.*

Algorithm 1:

Data : Two $\{\langle\rangle\}$-arc-annotated sequences S and T of length n and m, respectively
Result : true iff T occurs as an arc-preserving subsequence in S
begin

1 $S' = S[1 : \mathsf{FB}_S - 1] \cdot S[\mathsf{FB}_S : n]^R$
2 **foreach** *factorization* $T[\mathsf{LF}_T + 1 : \mathsf{FB}_T - 1]| = xy$ **do**
3 $T' = T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$
4 **if** T' *occurs as an arc preserving subsequence in* S' **then**
5 **return** *true*
6 **return** *false*
end

Proof. Consider Algorithm 1. Correctness of the algorithm follows from Lemma 1. What is left is to prove the time complexity. Clearly, $S' = S[1 : \mathsf{FB}_S - 1] \cdot S[\mathsf{FB}_S : n]^R$ is a $\{\sqsubset\}$-arc-annotated sequence. The key point is to note that, for any factorization $T[\mathsf{LF}_T + 1 : \mathsf{FB}_T - 1]| = xy$, the obtained $T' = T[1 : \mathsf{LF}_T] \cdot x \cdot (y \cdot T[\mathsf{FB}_T : m])^R$ is a $\{\sqsubset\}$-arc-annotated sequence as-well. Now let k be the number of arcs in T. So there are at most $m - 2k$ iterations to go before eventually returning false. According to the above, Line 4 constitutes an instance of APS($\{\sqsubset\}$,$\{\sqsubset\}$). But APS($\{\sqsubset\}$,$\{\sqsubset\}$) is a special case of APS($\{<, \sqsubset\}$,$\{<, \sqsubset\}$), and hence is solvable in $O(nm)$ time [6]. Then it follows that the algorithm as a whole runs in $O(nm(m - 2k)) = O(nm^2)$ time. □

Clearly, the proof of Theorem 3 relies on an efficient algorithm for solving APS($\{\sqsubset\}$,$\{\sqsubset\}$): the better the complexity for APS($\{\sqsubset\}$,$\{\sqsubset\}$), the better the complexity for APS($\{\langle\rangle\}$,$\{\langle\rangle\}$). We have used only the fact that APS($\{\sqsubset\}$,$\{\sqsubset\}$) is a special case of APS($\{<, \sqsubset\}$,$\{<, \sqsubset\}$). It remains open, however, wether a better complexity can be achieved for APS($\{\sqsubset\}$,$\{\sqsubset\}$). Theorem 3, combined with Observation 1, carries out easily to restricted versions.

Corollary 1. APS($\{\langle\rangle\}$,$\emptyset$) *is solvable in* $O(nm^2)$ *time.*

6 Conclusion

In this paper, we investigated the APS problem time complexity and gave a precise characterization of what makes the APS problem hard. We proved that APS(Crossing,Plain) is **NP**-complete thereby answering an open problem posed in [6]. Note that this result answers the last open problem concerning APS computational complexity with respect to classical complexity levels, *i.e.*, Plain, Chain, Nested and Crossing. Also, we refined the four above mentioned levels for exploring the border between polynomial time solvable and **NP**-complete problems. We proved that both APS($\{\sqsubset, \wr\}, \emptyset$) and APS($\{<, \wr\}, \emptyset$) are **NP**-complete and gave positive results by showing that APS($\{\wr\}$, $\emptyset$) and APS($\{\wr\}$,$\{\wr\}$) are polynomial time solvable. Hence, the refinement we suggest shows that APS problem becomes hard when one considers sequences containing $\{\wr, \alpha\}$-comparable arcs with $\alpha \neq \emptyset$. Therefore, crossing arcs alone do not imply APS hardness. It is of course a challenging problem to further explore the complexity of the APS problem, and especially the parameterized views, by considering additional parameters such as the cutwidth or the depth of the arc structures.

References

1. J. Alber, J. Gramm, J. Guo, and R. Niedermeier. Computing the similarity of two sequences with nested arc annotations. *Theoretical Computer Science*, 312(2-3):337–358, 2004.
2. G. Caetano-Anollés. Tracing the evolution of RNA structure in ribosomes. *Nucl. Acids. Res.*, 30:2575–2587, 2002.
3. P. Evans. *Algorithms and Complexity for Annotated Sequence Analysis*. PhD thesis, U. Victoria, 1999.
4. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
5. D. Goldman, S. Istrail, and C.H. Papadimitriou. Algorithmic aspects of protein structure similarity. In *Proc. of the 40th Symposium of Foundations of Computer Science (FOCS99)*, pages 512–522, 1999.
6. J. Gramm, J. Guo, and R. Niedermeier. Pattern matching for arc-annotated sequences. In *Proc. of the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS02)*, volume 2556 of *LNCS*, pages 182–193, 2002.
7. J. Guo. Exact algorithms for the longest common subsequence problem for arc-annotated sequences. Master's Thesis, Universitat Tubingen, Fed. Rep. of Germany, 2002.
8. T. Jiang, G.-H. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. In *Proc. 11th Symposium on Combinatorial Pattern Matching (CPM00)*, volume 1848 of *LNCS*, pages 154–165. Springer-Verlag, 2000.
9. V. Juan, C. Crain, and S. Wilson. Evidence for evolutionarily conserved secondary structure in the H19 tumor suppressor RNA. *Nucl. Acids. Res.*, 28:1221–1227, 2000.

10. G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *Proceedings of the 5th ACM International Conference on Computational Molecular Biology (RECOMB01)*, pages 193–202, 2001.
11. S. Vialette. On the computational complexity of 2-interval pattern matching. *Theoretical Computer Science*, 312(2-3):223–249, 2004.
12. K. Zhang, L. Wang, and B. Ma. Computing the similarity between RNA structures. In *Proc. 10th Symposium on Combinatorial Pattern Matching (CPM99)*, volume 1645 of *LNCS*, pages 281–293. Springer-Verlag, 1999.
13. M. Zuker. RNA folding. *Meth. Enzymology*, 180:262–288, 1989.

Performance Evaluation of Protein Sequence Clustering Tools

Haifeng Liu and Loo-Nin Teow

DSO National Laboratories (Kent Ridge), 27 Medical Drive, Singapore 117510
lhaifeng@dso.org.sg

Abstract. This paper aims to evaluate the clustering quality of various protein clustering tools that are publicly available as standalone applications. We first review the current protein sequence clustering methods, and introduce a new incrementally clustering tool denoted as PINC. We then propose an intuitive performance metric for evaluating them. The evaluation results of the tools on the public database Pfam are reported.

1 Introduction

With the enormous growth of public sequence database, grouping the sequences into protein families, is increasingly becoming important as it provides evolutionary, functional and structural information on the sequences. Protein families can be defined as those groups of molecules which share significant sequence similarity ([7]). Well characterized proteins within a family can hence allow one to reliably assign functions to family members whose functions are not known or not well understood. Many methods have been developed for clustering protein sequences according to similarity (or distance) information. This approach usually groups homologous proteins together via a similarity measure obtained by sequence comparison. However, correctly clustering a large number of protein sequences based on the approach still remains a huge challenge due to the following inherent characteristics of proteins:

- Multi-domain proteins. Proteins containing multiple domains that may be similar to different sets of unrelated clusters can confound the protein family detection methods and result in the incorrect grouping of proteins into families.
- Promiscuous domains. Promiscuous domains are widespread components of many proteins. Proteins assigned to a protein family purely on the basis of a promiscuous domain probably have very different functions with other members of that family.
- Fragmented proteins. Public databases may contain fragmented protein sequences (peptides). This incomplete information may lead to the incorrect assignment of a protein into a family.
- Distantly related proteins. Certain proteins sharing little sequence identity can have the same function. It is hard to group this kind of distantly related proteins together without comparison of their three-dimensional structures.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 877–885, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Although progress has been made on computational protein clustering in terms of accuracy and efficiency, very little work in the literature addresses the systematic comparison and evaluation of clustering results from different tools. Typically, different clustering algorithms yield different clustering solutions on the same data, and there is no agreed-upon guideline for choosing among them. In this paper, we first review the current protein sequence clustering methods, and introduce a new incrementally clustering tool denoted as PINC. We then conduct a performance assessment of PINC and selected representatives from a variety of computational protein clustering approaches on unified data sets using an intuitive performance metric proposed by us. The work would provide researchers a good understanding of current protein clustering tools, and assist the application of them in identifying structural and evolutionary relationships between proteins.

2 Clustering Methods

Current commonly employed protein clustering methods require sequence similarity relationships as input, but differ in dealing with the following major issues:

1. **Domain decomposition.** Basically, the protein clustering methods can be divided into two categories: the methods that have a preliminary decomposition of each protein into its constituent domains such as applied in [6, 9, 16], and the methods that directly cluster the protein sequence and are not domain aware including [1, 5, 8, 14] and so on. Although the former methods have achieved success in constructing well-known databases like PRODOM ([6]), ADDA ([12]) and so on, they hampered to a large extent by promiscuous domains, fragmented proteins, and proteins with complex domain structure. This paper focuses on the comparison of the latter methods.
2. **Sequence comparison manner.** Typically, most of algorithms such as [8, 9, 16, 12, 5] precompute or read in all pairwise sequence comparison. This all-against-all precomputation suffers from an excessive and unpredictable sequence comparison steps for large scale detection of protein families. On the other hand, several incremental clustering algorithms ([11, 14]) reduce the number of sequence comparisons by setting a representative sequence for each cluster. To assign a protein sequence into the existing clusters, the sequence is only compared to the representatives of the clusters.
3. **Sequence similarity measure.** As proposed in [14], the similarity between two protein sequences is first measured by the percentage of identical residues between them whereas most methods including [9, 8, 16, 5] measure the similarity between two protein sequence by the scores obtained from similarity search algorithms such as BLAST and FASTA ([17]).
4. **Cutoff threshold.** Almost all clustering algorithms set a cutoff threshold to adjust the granularity (or "tightness") of resulted clusters. Usually, the sequence similarity measures are directly adopted as the cutoff threshold

variables, such as E-value from BLAST ([16]) and FASTA score ([12]). Unfortunately, there is no single cutoff score that works for all families and different families may be formed at different cutoff scores. Other clustering criteria are adopted to solve the problem, such as in ([8]) the *power coefficient* of a stochastic matrix is varied to control the tightness of clusters. Recently a metric, called *cluster utility*, that combines inter-cluster dissimilarity and intra-cluster similarity,e has also been proposed to guide the sequence clustering ([13]).

5. **Transitivity.** Transitivity relationship is used by those clustering algorithms that precompute sequence comparison to infer distantly related proteins. However it is not fully understood if transitivity always holds and whether transitivity can be extended infinitely ([5]). Typically, a multi-domain protein may connect two clearly unrelated proteins and assign them into the same family. In [12], the graph theory based algorithms avoid the problem by controlling specificity of clusters using criteria which set the shared regions among family members. MCL algorithm in [8] simulates random walks withing a graph representing sequence similarity and iteratively eliminates the inter-family similarities due to multi-domain proteins.

In this work, we only conduct the performance evaluation to the programs that are publicly available as standalone applications. We briefly introduce them as follows: (we focus on the methods that automatically clustering the proteins without the detection of domains.)

BLASTCLUST. BLASTCLUST ([1]) automatically and systematically clusters proteins based on pairwise matches found using the BLAST algorithm. It finds pairs of sequences that have statistically significant matches and clusters them using single-linkage clustering which puts a sequence into a cluster if the sequence is a neighbor to at least one sequence in the cluster. The neighbor relationship between two sequences is determined by a score density threshold, whose adjustment also incurs the varied output clusters.

TRIBE-MCL. TRIBE-MCL ([8]) is a publicly available tool that aims to rapidly and accurately cluster protein sequences into families. It relies on the Markov cluster algorithm for the assignment of proteins into families based on precomputed all-against-all sequence similarity information using BLAST. The inflation value parameter ($I > 1$) of the MCL algorithm is used to control the granularity of clusters.

CD-HI. CD-HI ([14]) uses the same greedy incremental algorithm as in ([11]) where a sequence is compared with the existing representatives of clusters. If the similarity with any representative is above a given threshold, it is then included into that cluster. Otherwise a new cluster is created with the sequence as its representative. The program detects similar sequences by counting the number of identical short words between them, which speeds up the clustering process. The output of the program varies according to the identity threshold ($40-90\%$).

```
Input: Set P of protein sequences
       sequence similarity threshold (evalue) ε
Output: Set R of clusters, each cluster is a subset of P

R ← ∅;
while (P ≠ ∅)
    fetch a sequence x from P
    if (R = ∅)
        create a cluster C which contains x
        the seed of C ← x
        add C into R
    else
        compare x with the seeds of clusters in R
        find the nearest cluster O with seed s
        if similarity score between x and s is less than ε
            add x into O
            update the seed of O
        else
            create a cluster C which contains x
            the seed of C ← x
            add C into R
```

Fig. 1. Clustering algorithm PINC for detecting protein families

PINC. We propose an algorithm, denoted as PINC (Protein INcremental Clustering), which employs the similar incremental clustering algorithm as CD-HI, but uses BLAST to compare sequences as BLAST is a widely used sequence comparison tool and provides statistically significant alignments. The BLAST E-value score is directly used to vary the output of PINC. The details of the algorithm is shown in Fig. 1, whereby the *seed* of a cluster is a representative of sequences in the cluster. We adopt two options to determine the seed of a cluster. Firstly, to keep the algorithm simple, we determine the longest sequence in a cluster as its seed at present. We denote the algorithm as PINC-L. Alternatively, the seed of a cluster may be the consensus string of a multiple alignment of all sequences in the cluster. This approach, however, may require a large amount of additional computation and hence decrease the effectiveness of the algorithm. In our implementation, we use ClustalW ([19]) to do the multiple sequence alignments. The multiple alignment is updated whenever a new sequence is added into the cluster, either by aligning all the sequences together (version PINC-C1) or incrementally aligning the new added sequence with the existing alignment (version PINC-C2).

3 Benchmarking Data

Different protein sequence clustering programs have been evaluated on different data sets. CD-HI was reported to run fast on databases PDB, SWISS-PROT

([3]) and NR downloaded from the NCBI, but without any mention of its clustering quality. The effectiveness of TRIBE-MCL was tested in an extensive way. It was performed on SWISS-PROT and PDB databases, and the results were then respectively validated using the InterPro protein domain database ([2]) and the Structural Classification of Proteins (SCOP) ([15]) database. 1583 clusters (out of 1821 or 87%) obtained by running TRIBE-MCL on SWISS-PROT were found to have full correspondence with domains in InterPro, while 79% to 87% of proteins in clusters with SCOP classification are consistent with the SCOP cluster assignment. This validation process is not straightforward when there is a requirement of two or more reference databases being involved. In this paper, we evaluate the clustering tools by testing them on the subsets of Pfam database (release 10.0) of protein families ([4]) which contains 6,190 families and 553,048 protein sequences. Pfam families were constructed manually or semi-automatically using hidden Markov models (HMMs) based on the proteins in Swiss-Prot (41.10) and TrEMBL (23.15). Pfam families have permanent accession numbers and contain functional annotation and cross-reference to other databases, such as SCOP ([15]), and can be easily downloaded from its public website. The metrics for measuring the quality of clustering programs on the Pfam families are introduced in the next section.

4 Evaluation Metrics

To evaluate the comprehensive power of different clustering programs, we have separately applied the programs on the variable number of Pfam families. Given a set of testing protein families, after the clustering operation is performed, we say a family is *homogeneous* if its members are assigned into one resulted cluster, and a resulted cluster is *pure* if it only contains the proteins from one testing family. We quantitatively measure the *family homogeneity* and *cluster purity* as follows. Let N_f be the number of given testing protein families and N_c be the number of clusters produced by a tested clustering tool, for a family j having m_j members, its family homogeneity, denoted as FH_j, is defined as $FH_j = \sum_{i=1}^{N_c}(x_i/m_j)^2$ where x_i is the number of members of the family assigned into cluster i. Likewise, for a cluster i containing n_i proteins, its cluster purity, denoted as CP_i, is defined as $CP_i = \sum_{j=1}^{N_f}(y_j/n_i)^2$ where y_j is the number of members of the family j assigned into the cluster. The above definitions of family homogeneity and cluster purity are based on the complement form of the definition of *heterozygosity* for genetic markers which is $H = 1 - \sum {p_i}^2$ ([10])). Overall, the family homogeneity (FH) and the cluster purity (CP) obtained by the program are respectively defined as the weighted averages of all individual family homogeneities and all individual cluster purities as below:

$$FH = \sum_{j=1}^{N_f}(FH_j * m_j) / \sum_{j=1}^{N_f} m_j \tag{1}$$

and

$$CP = \sum_{i=1}^{N_c}(CP_i * n_i) / \sum_{i=1}^{N_c} n_i. \quad (2)$$

By analogy to the conventional F-measure ([18]) in the field of information retrieval, we measure the overall clustering quality by combining FH and CP as their harmonic mean:

$$F^* = 2 * FH * CP/(FH + CP). \quad (3)$$

Speed and memory are critical issues when clustering a large number of sequences given limited computer resources. All programs tested in this work run on a PC with dual Pentium III 1 Ghz CPU and 1 gigabyte physical memory which is operated by Red Hat 9.0. We also compare the running times of different programs.

5 Results

We have firstly run the selected programs on 1,710 protein sequences from 10 randomly chosen Pfam families. The results with different input parameters are shown in Table 1. Clearly, we observe that CD-HI runs very fast but with a poor performance due to a large number of fragmental clusters. This is because the primary goal of CD-HI is to reduce the redundancy of protein database, and it is more suitable for clustering the proteins that share certain sequence identity. The big differences between the F^* values obtained by BLASTCLUST with the varied parameter "density threshold (S)" settings imply that we may apply BLASTCLUST only when the input parameter can be carefully determined. PINC-L, PINC-C1 and PINC-C2 (PINCs) are comparable with TRIBE-MCL in terms of the clustering quality, with PINC-C1 achieving the highest F^* value of 0.793 when the input E-value threshold is set to 0.1. In addition, PINC-C1 and PINC-C2 outperform PINC-L by setting the seed of a cluster as the consensus string but at the cost of running speed. In particular, PINC-C1 has a significant drawback in terms of the running speed due to the large amount of multiple sequence alignments involved. Although PINCs run slower than TRIBE-MCL (the running time of TRIBE-MCL includes the time spent on the precomputation of sequence similarities using BLAST), we notice that the number of clusters by TRIBE-MCL is very sensitive to the (inflation) parameter value, whereas PINCs seem to be more robust against different (E-value) parameter values.

As TRIBE-MCL and PINCs outperform the other programs based on the above results, we prefer to utilize them in practice. Before that, we need to further measure their scalability with larger testing data set. We then run TRIBE-MCL and PINC-L on 28,117 protein sequences from 100 randomly chosen Pfam families. (We do not run PINC-C1 and PINC-C2 because they require a few weeks' running time in our PC). The results are shown in Table 2. TRIBE-MCL achieves a higher F^* value than PINC-L with the specific input parameter

Table 1. Evaluation results of protein clustering programs running on 10 Pfam families

Tools		Number of clusters	FH	CP	F^*	Running time (seconds)
CD-HI	$t = 0.4$	471	0.068	1	0.12	4
	$t = 0.6$	922	0.024	1	0.048	2
	$t = 0.9$	1,695	0.006	1	0.012	2
TRIBE-MCL	$I = 1.0$	2	0.999	0.183	0.310	60,860
	$I = 1.5$	11	0.770	0.661	0.712	530
	$I = 2.0$	383	0.374	0.890	0.526	472
PINC-L	$e = 0.01$	52	0.467	0.997	0.636	1,527
	$e = 0.1$	34	0.487	0.964	0.647	940
	$e = 1$	21	0.491	0.767	0.599	787
PINC-C1	$e = 0.01$	35	0.619	0.981	0.759	41,146
	$e = 0.1$	26	0.668	0.976	0.793	46,163
	$e = 1$	15	0.663	0.815	0.731	48,505
PINC-C2	$e = 0.01$	45	0.494	0.990	0.659	3,064
	$e = 0.1$	31	0.549	0.940	0.693	3,290
	$e = 1$	15	0.601	0.776	0.678	3,726

Table 2. Evaluation results of protein clustering programs running on 100 Pfam families

Tools		Number of clusters	FH	CP	F^*	Running time (seconds)
TRIBE-MCL	$I = 1.5$	325	0.587	0.909	0.713	64,792
	$I = 2.0$	7,448	0.195	0.952	0.324	49,180
PINC-L	$e = 0.01$	447	0.357	0.954	0.519	259,366
	$e = 0.1$	175	0.362	0.719	0.482	104,416
	$e = 1$	45	0.272	0.355	0.308	34,785

($I = 1.5$) while both programs run quite slowly (TRIBE-MCL requires longer than 2 weeks' running time with $I = 1$). However, we notice that TRIBE-MCL has a limitation on the input scale because of its precomputation of all-against-all sequence similarity and large amount of memory usage. In fact, TRIBE-MCL runs out of memory and crashes in our PC when we apply it to a clustering of 52,614 protein sequences from 300 Pfam families, whereas PINC-L achieves a F^* value of 0.496 after a run of 3 days. Therefore, for clustering large scale protein sequence sets, TRIBE-MCL should be considered only when abundant computing resources are available, while PINC can give reasonably good clustering accuracies but a long running time with limited computing resources.

6 Conclusions

Our evaluation results show that existing tools solely based on sequence similarity between proteins are still deficient in clustering quality and efficiency,

especially when coping with a large number of proteins, and suggest that more powerful tools need to be developed. In our future work, in order to improve the running speed of PINCs which currently hinders applying PINCs in large scale clustering, we will rewrite the programs in C language in place of current Perl scripts. In addition, we consider developing a parallel version of PINCs (or other programs) for dealing with the large scale clustering job with parallel computers.

References

1. S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, and et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
2. R. Apweiler, T.K. Attwood, A. Bairoch, A. Bateman, E. Birney, and et al. The InterPro database, an integrated documentation resoruce for protein familes, domains and functional sites. *Nucleic Acids Research*, 29:37–40, 2001.
3. A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Research*, 27:49–54, 1999.
4. A. Bateman, E. Birney, L. Cerruti, R. Durbin., and et al. The Pfam protein familes database. *Nucleic Acids Research*, 30(1):276–280, 2002.
5. E. Bolten, A. Schlipe, S. Schneckener, D. Schomburg, and R. Schrader. Clustering protein sequences — structure prediction by transitive homology. *Bioinformatics*, 17(10):935–941, 2001.
6. F. Corpet, F. Servant, J. Gouzy, and D. Kahn. ProDom and ProDom-CG: tools for protein domain analysis and whole genome comparisons. *Nucleic Acids Research*, 28(1):267–269, 2000.
7. M. O. Dayhoff. The origin and evolution of protein superfamilies. *Fed. Proc.*, 35:2132–2138, 1976.
8. A. J. Enright, S. V Dongen, and C. A. Ouzounis. An efficient alogrithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):276–280, 2002.
9. A. J. Enright and C. A. Ouzounis. GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16(5):451–457, 2000.
10. H. R. Gregorius. The concept of genetic diversity and its formal relationship to heterozygosity and genetic distance. *Math. Biosciences*, 41:253–271, 1978.
11. L. Holm and C. Sander. Removing near-neighbour redundancy from large protein sequence collections. *Bioinformatics*, 14:423–429, 1998.
12. S. Kim. BAG: A graph theoretic sequence clustering algorithm. In *http://bio.informatics.indiana.edu/sunkim/BAG*, School of Informatics, Indiana University-Bloomington.
13. S. Kim and A. Gopu. Cluster Utility: A new metric to guide sequence clustering. In *http://bio.informatics.indiana.edu/sunkim/BAG*, School of Informatics, Indiana University-Bloomington.
14. W. Li, L. Jaroszewski, and A. Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, 2001.
15. G. Murzin, A., S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol*, 247:536–540, 1995.
16. J. Park and S. A. Teichmann. DIVCLUS: an automatic method in the GEANFAMMER package that finds homologous domains in single- and multi-domain proteins. *Bioinformatics*, 14(2):144–150, 1998.

17. W. R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci.*, 85:2444–2448, 1988.
18. C.J. Van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
19. J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

A Data-Adaptive Approach to cDNA Microarray Image Enhancement

Rastislav Lukac[1], Konstantinos N. Plataniotis[1],
Bogdan Smolka[2⋆], and Anastasios N. Venetsanopoulos[1]

[1] The Edward S. Rogers Sr. Dept. of Electrical and Computer Engineering,
University of Toronto, 10 King's College Road, Toronto, M5S 3G4, Canada
{lukacr, kostas, anv}@dsp.utoronto.ca
http://www.dsp.utoronto.ca/∼lukacr
[2] Polish-Japanese Institute of Information Technology,
Koszykowa 86 Str, 02-008 Warsaw, Poland
bsmolka@ia.polsl.gliwice.pl

Abstract. A data-adaptive approach for cDNA microarray image enhancement is presented. Through the weighting coefficients adaptively determined from local microarray image statistics, the proposed technique tunes the overall filter's detail-preserving and noise-attenuating characteristics and uses both the spatial and spectral correlation of the cDNA image during processing. Noise removal is performed by tuning a membership function which utilizes the aggregated absolute differences between the cDNA microarray inputs localized within a processing window sliding over the image.

1 Introduction

cDNA microarray technology [1, 2, 3] is an advanced tool used to analyze changes in genome-wide patterns of gene expression in different populations of cells. Based on the widely used two-color Cy3/Cy5 system [4, 5], the control DNA sequence is usually fluor-tagged with green (Cy3), whereas the experimental sequence of the patient is tagged with red (Cy5). The composed Red-Green (RG) image (*Fig.1*) contains red, green and yellow spots which indicate that a particular gene is expressed in the control channel, experimental channel or both of them, respectively [5]. Each spot contains multiple copies of a single DNA sequence. The spots occupy a small fraction of the image area and they have to be individually located and isolated from the image background prior to the estimation of its mean intensity [1, 2, 6].

Although recognition of spots in either control or experimental channels seems to be straightforward, the task is complicated and challenging. Variation in image background and image artifacts, as well as spot sizes and positions

⋆ This research has been supported by a grant No. **PJ/B/01/2004** from the Polish - Japanese Institute of Information Technology.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 886–893, 2005.
© Springer-Verlag Berlin Heidelberg 2005

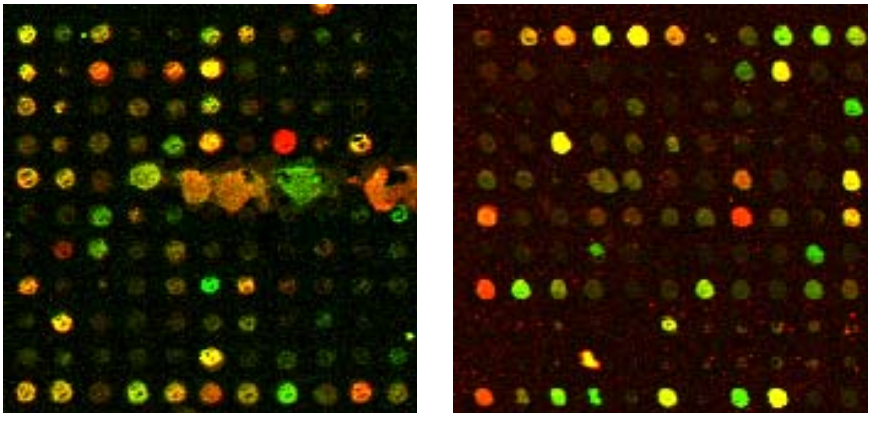


Fig. 1. cDNA microarray images

represent the major sources of uncertainty in spot finding and gene expression determination [5, 7]. In particular, non-specifically bounded DNA or dye molecules and the natural fluorescence of the glass slide result to a substantial noise floor in the microarray image. Since any errors or noise introduced usually limit the performance of the essential analysis procedure, if not defeat it altogether, image processing is necessary in order to eliminate processing errors from propagating further down the processing pipeline to the gene expression analysis tasks [1, 2, 6, 8]. Due to thousands of spots, the microarray image enhancement procedure, such as the one presented in this paper, should be fully automated.

2 Problem Formulation

Let us consider, a $K_1 \times K_2$ image $\mathbf{x} : Z^2 \rightarrow Z^2$ representing a two-dimensional matrix of two-component samples $\mathbf{x}_{(r,s)} = [x_{(r,s)1}, x_{(r,s)2}]$ with $r = 1, 2, ..., K_1$ and $s = 1, 2, ..., K_2$ denoting the image rows and columns, respectively. The component $x_{(r,s)1}$ indicates the R channel while $x_{(r,s)2}$ indicates the G channel. The two channels are combined to form the cDNA vector $\mathbf{x}_{(r,s)}$ in a two-dimensional vector space [2, 5]. Thus, the microarray data form a two-channel, Red-Green (RG) image [9, 10].

The cDNA microarray image suffers from high level of noise and edge uncertainty [7]. A number of noise sources mostly in the form of photon noise, electronic noise, laser light reflection and dust on the slide contribute to impairments and defects [2]. These imperfections along with the background fluorescence introduce into the image considerable variability in intensity both within and between the individual spots, as shown in *Fig.1.* It has been observed that changes of the pixel intensities from the foreground to the background can be attributed to the Gaussian nature of noise [4]. Isolated discrete artifacts and

outliers present in the cDNA microarray image can be attributed to impulsive noise [5]. In conventional image processing applications, noise corruption of this nature is most often modelled through a mixture of additive Gaussian noise and impulsive noise [11]. Therefore, the acquired cDNA signal $\mathbf{x}_{(r,s)} = [x_{(r,s)1}, x_{(r,s)2}]$ can be expressed through an additive noise model as follows [5]:

$$\mathbf{x}_{(r,s)} = \mathbf{o}_{(r,s)} + \mathbf{v}_{(r,s)} \tag{1}$$

where (r, s) indicates the spatial position of the samples in the image array. The vector $\mathbf{o}_{(r,s)} = [o_{(r,s)1}, o_{(r,s)2}]$ represents the original, noise-free cDNA signal, while the vectorial samples $\mathbf{v}_{(r,s)} = [v_{(r,s)1}, v_{(r,s)2}]$ are considered random in nature and are used to denote the various image impairments introduced during processing.

3 Proposed Method

Since the noisy samples deviate from other samples in a given data population [5, 11], we may consider uniformity in cDNA pixels inside a localized image area as the objective for enhancing the cDNA microarray image. Due to the various image impairments introduced during microarray image formation, there is no method to objectively determine whether or not a cDNA sample is noisy. Therefore, the proposed microarray image enhancement solution utilizes local image statistics of the cDNA samples localized inside a processing window. The window, defined as $\Psi_{(r,s)} = \{\mathbf{x}_{(i,j)}; (i,j) \in \zeta\}$, for $p = 1, 2, ..., K_1$ and $q = 1, 2, ..., K_2$, slides over the entire image $\mathbf{x}$ placing, successively, every pixel at the center of a local neighborhood denoted by ζ. The procedure replaces the cDNA sample $\mathbf{x}_{(r,s)}$ located at the window center (r, s) with the output $\mathbf{y}_{(r,s)} = f(\Psi_{(r,s)})$ of a filter function $f(\cdot)$ operating over the samples listed in $\Psi_{(r,s)}$.

The filter function $f(\cdot)$ and the area of support ζ can have numerous forms, such as those listed in [11]. Both $f(\cdot)$ and ζ determine the complexity and the overall performance of the procedure. A 3×3 rectangular shape window with $\zeta = \{(r-1, s-1), (r-1, s), ..., (r+1, s+1)\}$ is commonly used in image processing. Due to its versatility and demonstrated good performance, the same window shape is utilized throughout this paper. It has been proven that the conventional approaches such as component-wise median filter (MF) [12] and vector median filter (VMF) [13] produce robust estimates [5, 11]. However, noise and other impairments present in the acquired cDNA microarrays usually prohibit to accurately recognize between noise and edge pixels. Therefore, we will enhance the CDNA images using a fuzzy logic based smoothing approach (*Fig.2*) which is capable of tuning the overall filtering characteristics by employing a membership function defined over local image statistics [9].

To preserve the spectral characteristics of the cDNA microarray image, the proposed method constitutes the Red/Green (R/G) ratios from the cDNA vectorial inputs. The color-ratio approach takes advantage of the expected relative uniformity of the local R/G ratios. Due to the decreased high-frequency portion

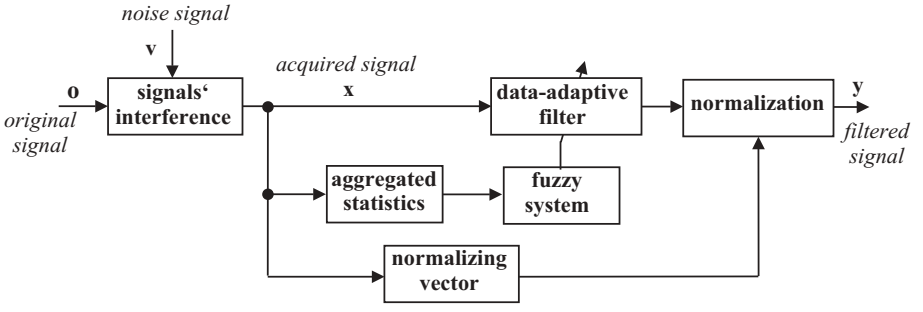


Fig. 2. Data-adaptive cDNA microarray image enahancer

of the signal, color-ratio based processing is able to preserve edges and structural content of the cDNA image better than the conventional approaches operating in the intensity domain. Each sample $\mathbf{x}_{(i,j)}$ is used to produce the Red/Green color ratio $x_{(i,j)1}/x_{(i,j)2}$. The produced ratio-values $x_{(i,j)1}/x_{(i,j)2}$, for $(i,j) \in \zeta$, are used to determine local image statistics, expressed here in the form of aggregated absolute differences between color ratios [9]:

$$D_{(i,j)} = \sum_{(g,h)\in\zeta} \left| x_{(i,j)1}/x_{(i,j)2} - x_{(g,h)1}/x_{(g,h)2} \right| \tag{2}$$

Following the processing pipeline depicted in *Fig.2*, the scalar values $D_{(i,j)}$, for $(i,j) \in \zeta$, are used to determine the fuzzy weights $\mu_{(i,j)}$ using the popular sigmoidal membership function [14] as follows:

$$\mu_{(i,j)} = \beta \left(1 + \exp\left\{D_{(i,j)}\right\}\right)^{-r} \tag{3}$$

where r is used to tune the weighting effect of the membership function and β is a normalizing constant. Note that each fuzzy weight $\mu_{(i,j)}$, for $(i,j) \in \zeta$, is associated with the input cDNA sample $\mathbf{x}_{(i,j)}$.

Since the most commonly used method to decrease the level of random noise present in the signal is smoothing, an averaging operation is required in order to replace the noisy vector $\mathbf{x}_{(r,s)}$ at the window center with a suitable vector representative for the local image area $\Psi_{(r,s)}$. Using the color ratios $x_{(i,j)1}/x_{(i,j)2}$ and the defuzzified weighting coefficients $w_{(i,j)} = \mu_{(i,j)} / \sum_{(g,h)\in\zeta} \mu_{(g,h)}$, the output vector $\mathbf{y}_{(r,s)} = [y_{(r,s)1}, y_{(r,s)2}]$ is obtained as follows [9]:

$$y_{(r,s)1} = x^*_{(r,s)2} \sum_{(i,j)\in\zeta} \left\{ w_{(i,j)} x_{(i,j)1}/x_{(i,j)2} \right\} \tag{4}$$

$$y_{(r,s)2} = x^*_{(r,s)1} \left[\sum_{(i,j)\in\zeta} \left\{ w_{(i,j)} x_{(i,j)1}/x_{(i,j)2} \right\} \right]^{-1} \tag{5}$$

where $\mathbf{x}^*_{(r,s)} = [x^*_{(r,s)1}, x^*_{(r,s)2}]$ is a vector, whose components are used to normalize the output (smoothed) color-ratio value $\sum_{(i,j)\in\zeta}\{w_{(i,j)}x_{(i,j)1}/x_{(i,j)2}\}$ to the individual intensities $y_{(i,j)1}$ and $y_{(r,s)2}$ corresponding to the recovered R and G color channels, respectively. The normalization vector $\mathbf{x}^*_{(r,s)}$ can be considered as equivalent to a robust estimate which statistically represents the input set $\Psi_{(r,s)}$. Using robust order-statistics principle, a normalization factor $\mathbf{x}^*_{(r,s)}$ is defined here as the component-wise MF [12]. Thus, the data-adaptive filter combines the fuzzy weighted average operations and the nonlinear median operations to normalize the output of the proposed microarray image enhancer.

4 Experimental Results

A number of cDNA microarray images have been used to evaluate the proposed filtering framework. Examples are shown in *Fig.3a* and *Fig.4a*. These images have

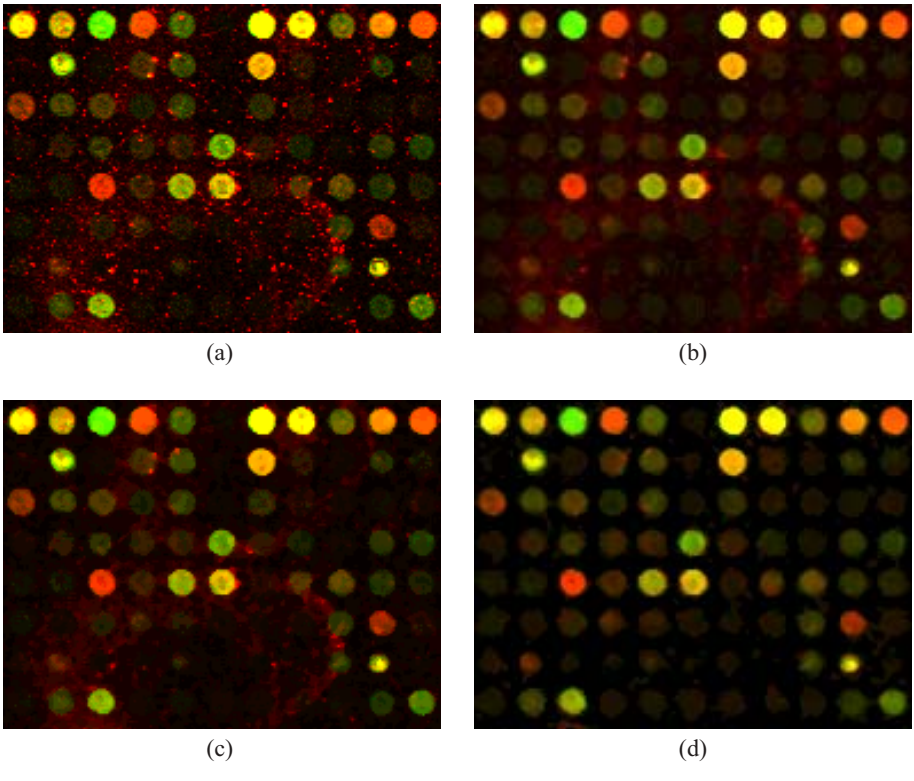


Fig. 3. Experimental results: (a) acquired image, (b) component-wise MF output, (c) VMF output, (d) the proposed method output

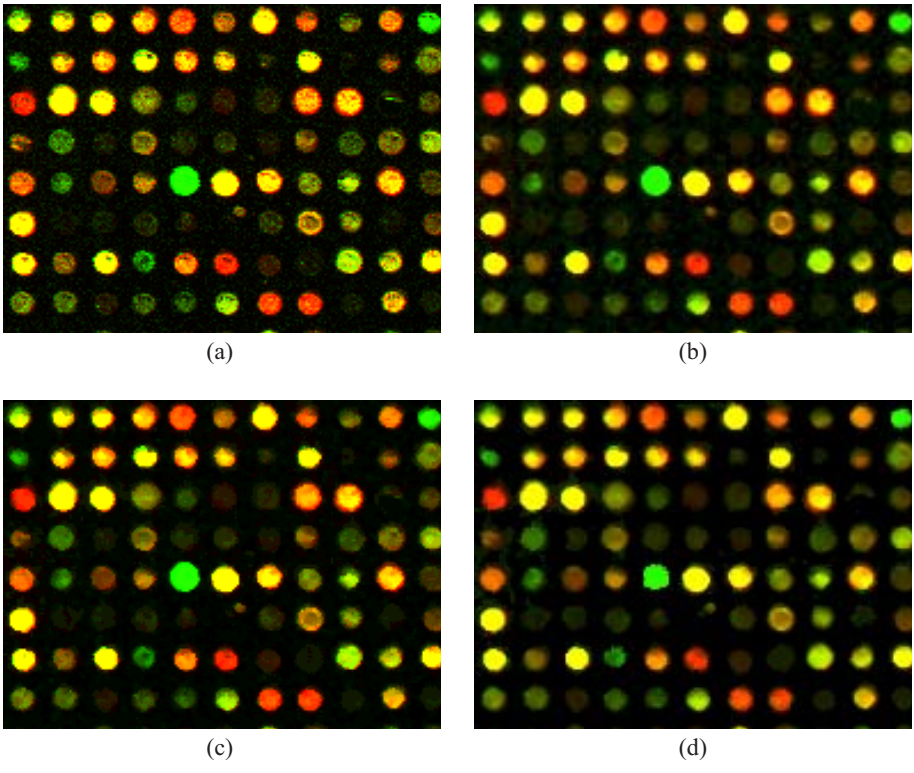


Fig. 4. Experimental results: (a) acquired image, (b) component-wise MF output, (c) VMF output, (d) the proposed method output

been captured using laser microscope scanners. The images vary in complexity and noise appearance. Note that all filtering results presented in this paper were obtained with a 3×3 square window.

The noise attenuation properties and detail-preserving capability of the proposed method are compared, in terms of performance, with the most popular and robust image filters such as the component-wise median filter (MF) [12] and the VMF technique [13].

Figs.3-4 show the acquired cDNA microarray images (a) and the denoised images (b-d) obtained using the filtering techniques under consideration. The result indicate that the proposed scheme achieves excellent balance between signal-detail preservation and noise attenuation. It can be seen that the well-known MF and VMF schemes as well as the proposed method remove well foreground noise such as the various impulses and other high-frequency image impairments. However, only the proposed method is capable of removing strong varied background intensities introduced in course of cDNA microarray technology. Similar

conclusions can be drawn by visually inspecting the 3-D plots of cDNA microarray images shown in *Fig.5*. This suggests that the proposed method should be used as a preprocessing tool for cDNA microarray spot localization.

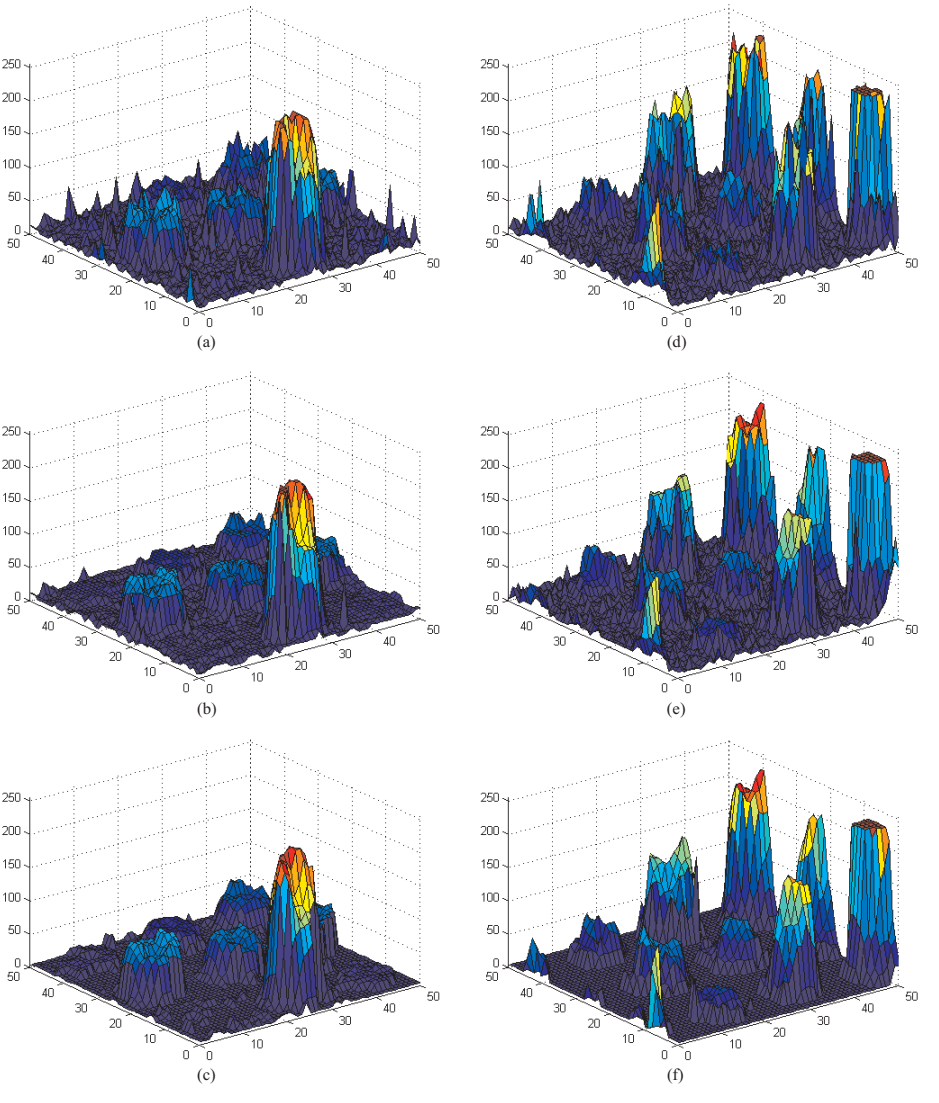


Fig. 5. 3-D visualization of cDNA spots: (a,d) acquired image, (b,e) VMF output, (c,f) the proposed method output

5 Conclusion

A new cDNA microarray processing tool has been introduced. The proposed method uses principles of fuzzy logic in conjunction with data-adaptive filtering concept defined over cDNA image data. Experimental results indicate that the proposed method is capable of removing noise in cDNA microarray images while preserving required data features for subsequent processing tasks.

References

1. Zhang, X.Y., Chen, F., Zhang, Y.T., Agner, S.G., Akay, M., Lu, Z.H., Waye, M.M.Y., Tsui, S.K.W.: Signal processing techniques in genomic engineering. Proceedings of the IEEE **90** (2002) 1822-1833
2. Lukac, R., Plataniotis, K.N.: cDNA microarray image segmentation using root signals. IEEE/ACM Transactions on Computational Biology and Bioinformatics, submitted (2004)
3. Eisen, M.B., Brown, P.O.: DNA arrays for analysis of gene expression. Methods in Enzymology **303** (1999) 179–205
4. Nagarajan, R.: Intensity-based segmentation of microarrays images. IEEE Transactions on Medical Imaging **22** (2003) 882–889.
5. Lukac, R., Plataniotis, K.N., Smolka, B., Venetsanopoulos, A.N.: A multichannel order-statistic technique for cDNA microarray image processing. IEEE Transactions on Nanobioscience **3** (2004) 272–285
6. Arena, P., Bucolo, M., Fortuna, L., Occhipinty, L.: Cellular neural networks for real-time DNA microarray analysis. IEEE Engineering in Medicine and Biology **21** (2002) 17–25
7. Wang, X.H., Istepian, R.S.H., Song, Y.H.: Microarray image enhancement using stationary wavelet transform. IEEE Transactions on Nanobioscience **2** (2003) 184–189
8. Dopazo, J.: Microarray data processing and analysis. In Microarray Data Analysis II, ed. SM Lin and KF Johnson, Kluwer Academic (2002) 43–63
9. Lukac, R., Plataniotis, K.N., Smolka, B., Venetsanopoulos, A.N.: cDNA microarray image processing using fuzzy vector filtering framework. Journal of Fuzzy Sets and Systems: Special Issue on Fuzzy Sets and Systems in Bioinformatics, to appear (2005)
10. Whitchurch, A.K.: Gene expression microarrays. IEEE Potentials **21** (2002) 30–34
11. Lukac, R., Smolka, B., Martin, K., Plataniotis, K.N., Venetsanopulos, A.N.: Vector filtering for color imaging. IEEE Signal Processing Magazine; Special Issue on Color Image Processing **22** (2005) 74–86
12. Pitas, I., Venetsanopoulos, A.N.: Order statistics in digital image processing. Proceedings of the IEEE **80** (1992) 1892–1919
13. Astola, J., Haavisto, P., Neuvo, Y.: Vector median filters. Proceedings of the IEEE **78** (1990) 678–689
14. Plataniotis, K.N., Androutsos, D., Venetsanopoulos, A.N.: Adaptive fuzzy systems for multichannel signal processing. Proceedings of the IEEE **87** (1999) 1601–1622

String Kernels of Imperfect Matches for Off-target Detection in RNA Interference

Shibin Qiu and Terran Lane

Dept. of Computer Science, University of New Mexico,
Albuquerque, NM, 87131
{sqiu, terran}@cs.unm.edu

Abstract. RNA interference (RNAi) is a posttranscriptional gene silencing mechanism frequently used to study gene functions and knock down viral genes. RNAi has been regarded as a highly effective means of gene repression. However, an "off-target effect" deteriorates its specificity and applicability. The complete off-target effects can only be characterized by examining all factors through systematic investigation of each gene in a genome. However, this complete investigation is too expensive to conduct experimentally which motivates a computational study. The sequence matching between an siRNA and its target mRNA allows for mismatches, G-U wobbles, and the secondary structure bulges, in addition to exact matches. To simulate these matching features, we propose string kernels measuring the similarity between two oligonucleotides and develop novel efficient implementations for RNAi off-target detection. We apply the algorithms for off-target errors in *C. elegans* and human.

1 Introduction

In addition to its use as a powerful tool to study gene functions and knockdown viral genes, RNA interference (RNAi) [1] also has the potential for validating drug design and treating diseases. RNAi is initiated by short interfering RNA (siRNA) of about 21 nt in length, either generated from double stranded RNA (dsRNA) by the enzyme Dicer, or delivered experimentally. Target mRNA transcripts that are matched with the siRNA are destroyed by the silencing complex RISC. RNAi has been regarded as a highly effective means of gene repression [2]. However, the effectiveness of RNAi can be compromised by the off-target knockdown, also known as nonspecific, cross-reactive, and false positive knockdown, which is defined as gene silencing inadvertently induced when a true target gene is intended to be knocked down. Characterizing the specificity of RNAi is critical to fully applying this novel technology to therapeutics [3, 4].

Examinations of RNAi off-target effect have yielded mixed results. The predominant opinion is that RNAi is highly specific to its target [2, 5]. But significant non-specific gene knockdowns have also been reported [6, 7]. In addition, some RNAi applications did not evaluate off-target effects [8]. To thoroughly understand RNAi specificity, off-target effect of silencing each gene should be

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 894–902, 2005.
© Springer-Verlag Berlin Heidelberg 2005

evaluated in a genome by considering multiple factors. However, it is expensive to conduct this comprehensive study experimentally. Computational approach is inexpensive to implement and able to extend experimental parameters into wider ranges and simulate what is impossible under normal biological conditions.

If an RNAi experiment assesses its specificity, the recommended procedure is using BLAST [9] searches to find off-target genes [10]. The sequence binding between an siRNA and its target mRNA allows for mismatches, G-U wobbles, and the secondary structure bulges [7]. Though BLAST allows for deletion, insertion and mismatch based on alignment, it cannot control exact patterns of the imperfect matches, such as their positions and lengths. To simulate siRNA-target matching, we develop string kernels that accurately generate matching patterns by controlling the pattern length and position.

Related to RNAi, computational methods exist to predict microRNA genes and targets [11, 12], putative RNAi [13], and siRNA efficacy [14]. To our knowledge, related works have not studied RNAi specificity. Our string kernels are the first of their kind to simulate siRNA and target matching that allows for accurate pattern control. We measure the similarity between two oligonucleotides with inner products in the feature space, similar to [15]. Leslie et al. [15] constructed a mismatch tree for computing their mismatch string kernel for a support vector machine classifier to detect protein families. But this mismatch kernel did not control mismatch positions. Amir et al. [16] developed an algorithm for string searches with single letter mismatch, which is not enough for RNAi.

We first define off-target error rate using information retrieval theory. We then propose novel string kernels to simulate siRNA and its target binding and develop efficient implementations for genome-wide computation. We study the dependencies of non-specificity on factors derived from RNAi experiments including dsRNA length, siRNA length, lengthes and positions of imperfect mismatches. Experiments are performed on *C. elegans* and human genomes.

2 String Kernels and Their Implementations

We first define the exact match string kernel and the off-target error rate. We then present the imperfect match kernels and their implementations.

Sequence matching between an siRNA from a given gene and another endogenous gene signifies an occurrence of off-target knockdown, which is a typical setting for gene function analysis [8]. For viral gene silencing, we conceptually combine the viral gene into the host genome and use the same setting.

2.1 Matching siRNA and Target with the Exact Match String Kernel

We describe each gene by its possible contiguous subsequences of length n (17-28 nt), called n-mers, or n-grams, representing siRNAs. Gene g_x in the input space $\mathcal{X}$, consisting of sequences drawn from the alphabet $\mathcal{A} = \{a, c, g, t\}$, is mapped into an n-gram feature space $\mathbb{R}^{4^n}$ by the feature map of exact match,

$$\Phi^{ex}(g_x) = (\phi_a(g_x))_{a \in \mathcal{A}^n}, \tag{1}$$

where $\phi_a(g_x)$ is the number of times n-gram a occurs in g_x. Therefore, the image of g_x is the coordinates in the feature space indexed by the number of occurrences of its constituent n-mers. A gene g_y is said to match g_x if the following is true,

$$K(g_x, g_y) = \langle \Phi^{ex}(g_x), \Phi^{ex}(g_y) \rangle \geq T, \tag{2}$$

for a threshold T. The similarity measure $K(g_x, g_y) = \langle \Phi^{ex}(g_x), \Phi^{ex}(g_y) \rangle$ in (2) defines a kernel as used for a support vector machine classifier [17]. We use the kernel to match an siRNA and its target, instead of classifying. Since any match between an siRNA and an mRNA will silence the gene, we choose $T = 1$.

Implementation of the Exact Match Kernel. Computing the similarity of (2) directly in a vector space requires $O(DF4^n)$ time, where F is the number of n-grams in the genome (40×10^6 for *C. elegans* and 60×10^6 for human) and D (close to F) is the amount of n-mers to be compared in the coding sequences. For genome-wide scan, this computing time is prohibitive and can be improved since the the feature vectors are sparse. We use an *inverted file* where the n-mers serve as identifiers and their gene names and positions within the genes serve as attributes. We use the positions for imperfect matches later. If we ignore n-mers having zero occurrence and allow for duplicate n-mers, a gene g_x can be represented in the feature space compactly,

$$\Phi^{ex}(g_x) = \{(a_1, p_1), (a_2, p_2), ..., (a_{k_x}, p_{k_x})\}, \tag{3}$$

where a_j, $1 \leq j \leq k_x$, is the jth n-gram of g_x, p_j is its position on g_x, and k_x is the number of n-mers in g_x. In the inverted file, the records for g_x contains the triples $\langle a_1, g_x, p_1 \rangle$, $\langle a_2, g_x, p_2 \rangle$, $\langle a_3, g_x, p_3 \rangle$,...,$\langle a_{k_x}, g_x, p_{k_x} \rangle$. The inverted file for a genome is the collection of the triples of its genes. To speed up computation, we sort the inverted file on the n-mer field using a binary search tree (BST).

$K(g_x, g_y)$ in (2) is computed by searching each n-mer of g_x for g_y in the inverted file. $K(g_x, g_y)$ is the number of occurrences of g_y among the matched genes. Each search in the BST takes $O(\log F)$ time, resulting in a time of $O(k_x \log F)$ for computing $K(g_x, g_y)$.

2.2 Definition of Off-target Error Rate

We define the off-target error using the exact match kernel, but it is the same for other kernels. To simulate Dicer's cleavage of dsRNA (100-400bp) into siRNAs, we take an oligonucleotide o_x, as dsRNA, from gene g_x and map it into the feature space using the exact match kernel. Expressed compactly as in (3), $\Phi^{ex}(o_x) = \{(s_1, p_1), (s_2, p_2)..(s_{l_x}, p_{l_x})\}$, where s_j is the jth n-mer in o_x. To obtain the matched genes based on (2), we compute the kernel $K(o_x, g_y)$ for each gene g_y, for $1 \leq y \leq G$, where G is the total number of genes in a genome. These calculations use BST searches, as described previously.

Let $C_x = \{g_{x_1}, g_{x_2}, ...\}$ be the set of genes whose kernel values with o_x satisfies (2), excluding g_x itself. The *precision* of a search is the proportion of correct documents to the total number of documents matched. Here, only g_x is correct and the number of genes returned is $1 + |C_x|$. So the precision of g_x is $P_x =$

$1/(1+|C_x|)$. We define the *off-target error* of g_x as $E_x = 1-P_x = |C_x|/(1+|C_x|)$. We define the average error rate as,

$$E(\Theta) = \sum_{i=1}^{G} E_i/G, \tag{4}$$

where E_i is the error for silencing gene g_i. And $\Theta = \langle l, n, m, p, b, q, w, r\rangle$ is the set of parameters, where l is the dsRNA length; n, siRNA length; m, mismatch length; p, position of the mismatch; b and w are lengths of the bugle and wobble; and q and r are positions of the bulge and the wobble, respectively. Figure 1 shows some of the parameters. The average number of incorrect genes targeted by silencing each gene in the genome is, $Z(\Theta) = E(\Theta)/(1 - E(\Theta))$.

Fig. 1. Exact match (left), mismatch (and wobble, middle), and bulge (right)

2.3 The Feature Maps of Mismatch, Bulge, and Wobble

For an n-mer a from an alphabet $\mathcal{A}$, define its mismatch neighborhood $N_{m,p}^{mis}(a)$ as all n-mer γ from $\mathcal{A}$ that differ from a by at most m contiguous mismatches starting at position p in a, and $\Phi_{m,p}^{mis}(a) = (\phi_\gamma(a))_{\gamma\in\mathcal{A}^n}$, where $\phi_\gamma(a) = 1$ if $\gamma \in N_{m,p}^{mis}(a)$, and $\phi_\gamma(a) = 0$, otherwise. The feature map of a gene g_x is defined as the sum of the feature maps of its n-mers,

$$\Phi_{m,p}^{mis}(g_x) = \sum_{a\in g_x} \Phi_{m,p}^{mis}(a). \tag{5}$$

The bulge neighborhood $N_{b,q}^{bulge}(a)$ for n-mer a is defined as all $(n+b)$-mers γ from the target that match a exactly everywhere except by a bulge of b nt long starting at position q on γ, and $\Phi_{b,q}^{bulge}(a) = (\phi_\gamma(a))_{\gamma\in\mathcal{A}^{n+b}}$, where $\phi_\gamma(a) = 1$ if $\gamma \in N_{b,q}^{bulge}(a)$, and $\phi_\gamma(a) = 0$, otherwise. The feature map of g_x is defined as the sum of the feature maps of its n-mers,

$$\Phi_{b,q}^{bulge}(g_x) = \sum_{a\in g_x} \Phi_{b,q}^{bulge}(a). \tag{6}$$

The wobble feature map $\Phi_{w,r}^{wobble}(.)$ is defined similarly to $\Phi_{m,p}^{mis}(.)$, except only G-U wobble is allowed in its neighborhood.

Note the mismatch, bulge and wobble neighborhoods are supersets of the exact match region. By defining the similarity neighborhood for the combination of mismatches, bulges, and wobbles as the union of the neighborhoods defined above, we can define the feature map of simultaneous mismatch, bulge and wobble $\Phi_{m,p,b,q,w,r}^{mbw}(.)$ accordingly. The string kernels are derived using inner products based on the above feature maps, as in (2).

2.4 Implementations of Mismatch, Bulge, and Wobble Kernels

Let $S = \{s_1, s_2...s_N\}$ be a set of strings of length k from an alphabet $\mathcal{A}$. Suppose string $s_i = a_1a_2..a_k$, where $a_j \in \mathcal{A}$, $1 \leq j \leq k$, has reverse string $\overline{s_i} = a_k a_{k-1}..a_1$.

Definition A *mirrored tree* of a binary search tree (BST) populated with strings from S is the BST populated with reverse strings $\overline{s_1}$, $\overline{s_2}$,...,$\overline{s_N}$. A u *leading range* of a string s from S searched in a BST is the set of nodes returned by a search that only matches the beginning u letters of s.

Algorithm 1. Mirrored Tree Search, MTS(n,m,p)

1: Build BST T_1 and mirrored BST T_2 for the inverted file using n-mer
2: **for each** gene g_i in the genome **do**
3: Take a subsequence d_i in g_i
4: **for each** n-mer s_j^i in d_i **do**
5: Get R_1, the $p-1$ leading range of s_j^i from T_1
6: Get R_2, the $n-m-p+1$ leading range of $\overline{s_j^i}$ from T_2
7: Find $C_j^i = R_1 \bigcap R_2$
8: **end for**
9: Calculate off-target error E_i for g_i using $C_i = \bigcup_j C_j^i$
10: **end for**
11: Calculate average off-target error for the genome

The mismatch kernel in (5) can be computed by the mirrored tree search (MTS) in Algorithm 1, whose correctness proof we omit due to space limitation. At step 5 and 6, the substring before the mismatch is exact-matched in T_1 and the leading range is stored in R_1, the substring after the mismatch is exact-matched in T_2 and the leading range is stored in R_2. The genes corresponding to the mismatch letters are sandwiched in C at step 7 by the intersection based on gene names and positions of the n-mers. At step 9, E_i is computed using C_i.

Let the size of the inverted file be F and the total number of n-grams from all dsRNAs be D. MTS has a cost of $T_{mt} = O(D(2\log F + C))$, where $O(2\log F)$ is the search time in the BSTs, and C is the cost of obtaining the leading ranges and the intersection. Using proper join algorithms, C can be bounded by $O(|R1| + |R2|)$ [18]. Empirically, C is small and treated as a constant. The mismatch tree algorithm [15] has a complexity of $O(DLn^m 4^m)$, where L is the average length of genes. Since usually $F < 10^8$ and $\log F < 30$, MTS is much faster. A straightforward way of mismatch search—done by searching each variant of the n-mer once—needs 4^m exact-match searches in the BST and consumes a time of $T_{sf} = O(4^m DlogF)$. MTS's speedup over the straightforward method is roughly, $Sp \approx O(4^m)$. Empirical results demonstrated that MTS achieved speedups of 2 orders of magnitude on average for the 2 organisms. However, it uses more space because of the mirrored tree.

The wobble kernel can be implemented by modifying Algorithm 1 to allow only G-U wobbles in step 7. The bulge kernel can also be computed by modifying

MTS. At step 1, use $(n+b)$-mer on the target; at step 4, elongate s_i and $\overline{s_i}$ to length $n+b$ by appending b arbitrary letters; get $q-1$ leading range R_1 from T_1 at step 5; get $n+b-q+1$ leading range R_2 from T_2 at step 6. Thus R_1 contains the genes corresponding to the exact matches on the first $q-1$ letters, leaving $n+b-(q-1)$ letters unconstrained in T_1. And R_2 contains the genes corresponding to exact matches on the last $n-q+1$ letters, leaving $b+q-1$ letters unconstrained in T_2. The intersection yields bulge matches of the $(n+b)$-mer on the target and n-mer of the siRNA. The combination of imperfect matches can also be computed using MTS. These implementations are efficient and flexible, allowing the imperfect match patterns to occur anywhere in the n-grams.

3 Experiments

The cDNA sequences of 22,168 genes of *C. elegans* (release WS110) were obtained from Sanger Institute. The human genes representing 27,852 mRNAs were taken from the RefSeq database at NCBI.

We show off-target error rates using an exact match in Fig. 2(a)(*H. sapiens*) and (b)(*C. elegans*). When siRNA length increases, the off-target error decreases. When an siRNA is short, a small length increase improves the error dramatically and the error rate curve is steep. But when the siRNA is relatively long, further length increase only improves the error slightly. The observed siRNA length in biological experiments is between 19 and 25 nt, which is consistent with our finding that increasing siRNA length beyond 25 does not improve specificity much. When siRNAs are short, longer dsRNAs generate larger errors. When they are relatively long, longer dsRNAs cause slight increases in the error. The error rate in human is higher than in *C. elegans*, but they present similar patterns. Therefore we focus on the *C. elegans* genome in the following. Fig. 2(c) plots the error rates when bulge, mismatch and wobble exist separately. Bulge barely increases off-target error. This is because the larger neighborhood caused by a bulge is cancelled by the reduced chances of target binding made by an increased length of the subsequence on the target to form the bulge. Wobble increases the error slightly, but its increase becomes insignificant for siRNAs longer than 21

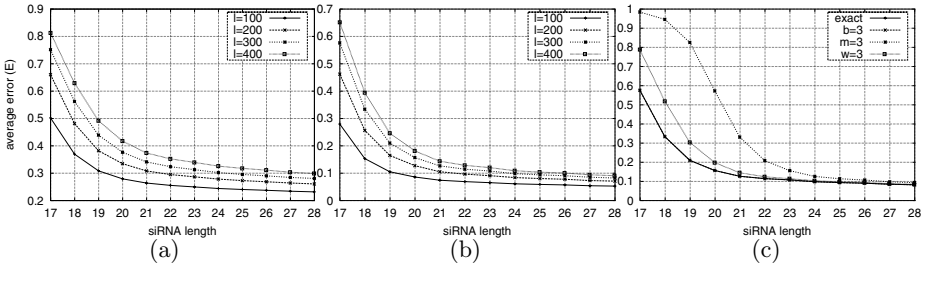


Fig. 2. Off-target error rates. (a)*H. sapiens*, exact match; (b)*C. elegans*, exact match; ($l = 100 - 400$ and $n = 17 - 28$). (c)*C. elegans* (l=300, $b = m = w = 3$)

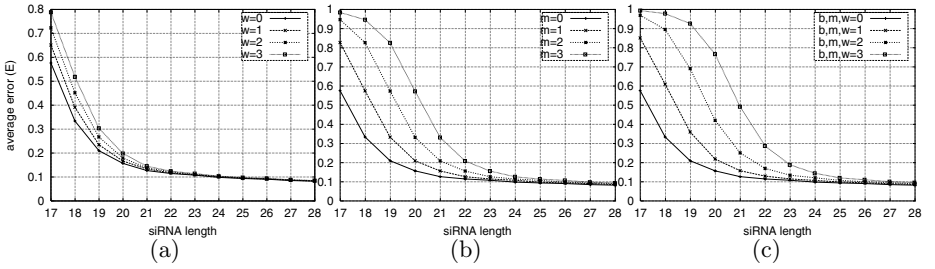


Fig. 3. Effects of imperfect matches in *C.* elegans (l=300). (a)wobble; (b)mismatch; (c) simultaneous mismatch, bugle, and wobble

nt. Mismatch caused a dramatic increase in error rate, but its effect diminished when $n \geq 25$. Results also show that positions of these imperfect matches did not change error rates. Therefore, position effects observed in biological experiments are due to thermodynamic properties of the sequences and RISC [7]. Fig. 3(a) and (b) show the effects of a range of m and w, suggesting, again, that mismatch is more critical than G-U wobble. Fig. 3(c) displays simultaneous mismatch, bugle, and wobble using a combined kernel, indicating that the increase in off-target errors did not diminish until $n \geq 25$.

$$\hat{E} = 18.87 - 0.58n + 0.22l + 51m + 10w - 0.008nl - 2nm - 0.4nw. \quad (7)$$

To predict off-target errors in *C. elegans*, we fitted a nonlinear regression function (7) using SAS software package, where $\hat{E} = \hat{E}(n, l, m, w, b)$ is in percents and length parameters are in nucleotides. As shown, n has a negative coefficient, consistent with off-target errors decreasing with n. The large coefficient with the m term shows that mismatch raises the error remarkably. The nonlinear terms are interactions of n with imperfect match parameters and also have negative coefficients, demonstrating the dominance of n over other parameters. Given a range of parameters, predictions using this function provide estimates of the off-target error rates and save computational effort.

4 Conclusions

Because the specificity of RNA interference is of fundamental importance and mixed results have been reported, we conducted a computational study of the off-target effects. To simulate the matching between an siRNA and its target mRNA, we introduced string kernels of imperfect matches including mismatches, bulges, and wobbles that are able to control the positions and lengths of the patterns. To improve computational performance for genome-wide scans, we proposed efficient implementations for the string kernels, which sped up computation

substantially. Results in *C. elegans* and human genomes indicated that off-target errors were substantial (5% to 80%). We found that bulge did not significantly increase off-target error, wobble raised the error slightly, and mismatch elevated the error dramatically. But the increase in error rates all diminished when siRNAs are long enough. Computationally, positions of the imperfect matches did not change off-target errors. Therefore, position-effects are due to thermodynamic properties of the components. A nonlinear regression function was fitted to predict off-target error rates. In summary, the off-target effect presents a real but not prohibitive issue and should be controlled in RNAi experiments.

Acknowledgement

This work is supported by NIH under grant number P20RR18754 from the Institutional Development Award Program of the National Center for Research Resource. We thank Coenraad M. Adema for his constructive suggestions.

References

1. Fire, A., Xu, S.Q., et al.: Potent and specific genetic interference by double-stranded RNA in *Caenorhabditis elegans*. Nature **391** (1998) 806–811
2. Elbashir, S.M., Martinez, J., et al.: Functional atonamy of siRNA for mediating efficient RNAi in *Drosophila melanogaster* embryo lysate. The EMBO Journal **20** (2001) 6877–6888
3. Dillin, A.: The specifics of small interfering RNA specificity. Proc. Nat. Acad. Sci. USA **100** (2003) 6289–6291
4. Check, E.: Hopes rise for RNA therapy as mouse study hits target. Nature **432** (2004) 136
5. Tuschl, T., Zamore, P.D., et al.: Targeted mRNA degradation by double-stranded RNA *in vitro*. Genes Dev. **13** (1999) 3191–3197
6. Jackson, A.L., Bartz, S.R., et al.: Expression profiling reveals off-target gene regulation by RNAi. Nature Biotechnology **21** (2003) 635–637
7. Saxena, S., Jonsson, Z.O., Dutta, A.: Small RNAs with imperfect match to endogenous mRNA repress translation. Journal of Biological Chemistry **278** (2003) 44312–44319
8. Kamath, R.S., Fraser, A.G., et al.: Systematic function analysis of the *C. elegans* genome using RNAi. Nature **421** (2003) 231–237
9. Altschul, S.F., et al.: Basic local alignment search tool. J. Mol. Biol. **215** (1990) 403–410
10. Elbashir, S.M., Harborth, J., et al.: Analysis of gene function in somatic mammalian cells using small interfering RNAs. Methods **26** (2002) 199–213
11. Lim, L.P., Glasner, M.E., et al.: Vertebrate microRNA genes. Science **299** (2003) 1540
12. Lewis, B.P., et al.: Prediction of mammalian microRNA targets. Cell **115** (2003) 787–798
13. Horesh, Y., Amir, A., et al.: A rapid method for detection of putative RNAi target genes in genomic data. Bioinformatics **19** (2003) ii73–ii80 Suppl. 2.

14. Chalk, A.M., Wahlestedt, C., Sonnhammer, E.L.: Improved and automated prediction of effective siRNA. Biochemical and Biophysical Research Comm. **319** (2004) 264–274
15. Leslie, C., Eskin, E., et al.: Mismatch string kernels for discriminative protein classification. Bioinformatics **1** (2003) 1–10
16. Amir, A., Landau, G., et al.: Text indexing and dictionary matching with one error. J. Algorithms **37** (2000) 309–325
17. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
18. Garcia-Molina, H., Ullman, J.D., Widom, J.D.: Database Systems: The Complete Book. Prentice Hall Inc., New Jersey (2002)

A New Kernel Based on High-Scored Pairs of Tri-peptides and Its Application in Prediction of Protein Subcellular Localization*

Zhengdeng Lei and Yang Dai**

Department of Bioengineering (MC063),
University of Illinois at Chicago,
851 South Morgan Street, Chicago, IL 60607, USA
{zlei2, yangdai}@uic.edu

Abstract. A new kernel has been developed for vectors derived from a coding scheme of the tri-peptide composition for protein sequences. This kernel defines the sequence similarity through a mapping that transforms a tri-peptide coding vector into a new vector based on a matrix formed by the high BLOSUM scores associated with pairs of tri-peptides. In conjunction with the use of support vector machines, the effectiveness of the new kernel is evaluated against the conventional coding schemes of k-peptide ($k \leq 3$) for the prediction of subcellular localizations of proteins in Gram-negative bacteria. It is demonstrated that the new method outperforms all the other methods in a 5-fold cross-validation.

Keywords: protein subcellular localization, Gram-negative bacteria, BLOSUM matrix, kernel, support vector machine.

1 Introduction

Advances in proteomics and genome sequencing are generating enormous numbers of genes and proteins. The development of automated systems for the annotation of protein structure and function has become extremely important. Since many cellular functions are compartmentalized in specific regions of the cell, subcellular localization of a protein is biologically highlighted as a key element in understanding its function. Specific knowledge of subcellular location can direct further experimental study of proteins.

Methods and systems have been developed during the last decade for the predictive task of protein localization. Machine learning methods such as Artificial Neural Networks, the k-nearest neighbor method, and Support Vector Machines (SVM) have been utilized in conjunction with various methods of feature

* This research is partially supported by National Science Foundation (EIA-022-0301) and Naval Research Laboratory (N00173-03-1-G016). The authors are thankful for Deepa Vijayraghavan for the assistant with computing environment.

** Corresponding author.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 903–910, 2005.
© Springer-Verlag Berlin Heidelberg 2005

extraction from protein sequences. Most of the early approaches employed the amino acid composition and the di-peptide composition [7, 11, 20] to represent sequences. This method may miss the information on sequence order and the interrelationships between the amino acids. In order to overcome this shortcoming, it has been shown that motifs, frequent-subsequences, and functional domains, which are obtained from various databases (SMART, InterPro, PROSITE) or extracted using Hidden Markov Models and data mining techniques, can be used for the representation of protein sequences for the prediction of subcellular localizations [2, 3, 6, 21, 22]. Methods have also been developed based on the use of the N-terminal sorting signals [1, 5, 9, 15, 17, 18, 19] and sequence homology searching [16].

Most robust methods adopt an integrative approach by combining several methods, each of which may be a suitable predictor for a specific localization or a generic predictor for all localizations. PSORT is an example of such a successful system. Developed by Nakai and Kanehisa [18], PSORT, recently upgraded to PSORT II [10, 17], is an expert system that can distinguish between different subcellular localizations in eukaryotic cells. It also has a dedicated subsystem PSORT-B for bacterial sequences [8].

Several recent studies [14, 23], however, have indicated that a predicting system based on the use of generalized k-peptide compositions or sequence homology could obtain similar or better performance compared to that of the integrated system PSORT-B. The outcome from our work also supports these findings.

In this study, a new similarity measurement for protein sequences has been developed based on the use of a matrix derived from high-scored pairs of tri-peptides. Each protein sequence is first coded by its tri-peptide composition. Since the repeating of the same tri-peptide is relatively lower comparing to that of di-peptides, the tri-peptide coding is more faithful in retaining the order of amino acids. Each pair of tri-peptides is then assigned with a score based on a BLOSUM matrix. A small portion of pairs with high scores is selected to retain their original scores in order to reduce noise and the computational time. The rest of pairs are given zero scores. The reassigned score associated with each pair of tri-peptides is then considered as an entry of an imaginary matrix D, which is named as the matrix of high-scored pairs of tri-peptides. It is obvious that pairs with more than two amino acids in common or sharing residues with high BLOSUM scores usually receive higher scores. Then each tri-peptide coding vector $\boldsymbol{x}$ is mapped to another vector $D\boldsymbol{x}$, and the similarity between the sequences is measured by those mapped vectors. That is, the kernel is defined based on these mapped vectors.

The new method is evaluated against the coding schemes of k-peptide ($k \leq 3$) compositions for the prediction of subcellular localizations for proteins obtained from Gram-negative bacteria [8]. It is demonstrated by the result of a 5-fold cross-validation that the new method outperforms the coding methods based on the k-peptide compositions.

2 Method

This section introduces the new kernel for the coding vectors derived from the tri-peptide compositions for protein sequences. The coding scheme based on the tri-peptide composition has been used in protein fold recognition [13], but has never been evaluated for the prediction of subcellular localizations. First a short description of support vector machines, the machine learning method used in this study will be presented.

2.1 Support Vector Machines

Suppose that a set of m training points $\boldsymbol{x}_i$ $(1 \leq i \leq m)$ in an n-dimensional space is given. Each point $\boldsymbol{x}_i$ is labeled by $y_i \in \{1, -1\}$ denoting the membership of the point. An SVM is a learning method for binary classification. Using a nonlinear transformation ϕ, it maps the data to a high dimensional feature space in which a linear classification is performed. It is equivalent to solving the quadratic optimization problem:

$$\begin{array}{ll} \min\limits_{\boldsymbol{w},b,\xi_1,\ldots,\xi_m} & \frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w} + C\sum\limits_{i=1}^{m}\xi_i \\ \text{subject to} & y_i(\phi(\boldsymbol{x}_i)\cdot\boldsymbol{w}+b) \geq 1-\xi_i \quad (i=1,\ldots,m), \\ & \xi_i \geq 0 \quad (i=1,\ldots,m), \end{array}$$

where C is a parameter. The decision function is defined as $f(\boldsymbol{x}) = \text{sign}(\phi(\boldsymbol{x})\cdot\boldsymbol{w}+b)$, where $\boldsymbol{w} = \sum_{i=1}^{m}\alpha_i\phi(\boldsymbol{x}_i)$ and α_i $(i=1,\ldots,m)$ are constants determined by the dual problem of the optimization defined above. Define a dot product $k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ for any pair of mappings $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$, i.e., $k(\boldsymbol{x}_i,\boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x}_j)$. This is called kernel function. The matrix $K_{ij} = k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ is called kernel matrix. The decision function can be represented as $f(\boldsymbol{x}) = \text{sign}(\sum_{i=1}^{m}\alpha_i\phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x}) + b) = \text{sign}(\sum_{i=1}^{m}\alpha_i k(\boldsymbol{x}_i,\boldsymbol{x}) + b)$. The typical kernel functions are, for example, polynomial kernel $(\boldsymbol{x}_i\cdot\boldsymbol{x}_j)^d$ $(d \geq 1)$ and Gaussian kernel $\exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$. For other details of SVMs refer to [4].

2.2 Sequence Coding Schemes and a New Kernel Based on High-Scored Pairs of Tri-Peptides

The coding schemes of protein sequences based on k-peptide compositions or their variations have been demonstrated effective in the predictions of protein folds and subcellular localizations, in conjunction with the use of machine learning tools such as neural networks and support vector machines [16, 23]. If $k = 1$, then the k-peptide composition reduces to the amino acid composition, and if $k = 2$, the k-peptide composition gives the di-peptide composition. When k becomes larger, the k-peptide compositions will cover more global sequence information, but at the same time, such a coding scheme becomes less attractive from a computational viewpoint.

In order to code a sequence, a window with a length of k is moved along the sequence from the first amino acid to the kth amino acid from the end. Every

k-letter pattern that appears in the window is recorded with increment of 1 in the corresponding entry of the vector. Upon the termination of this procedure, the vector provides the k-peptide composition of the sequence. The final vector is normalized by dividing the number of window positions associated with that sequence. Since the symbol "X" may appear in some sequences, it is added to the set of the original 20 symbols of the amino acids to give a total of 21. Therefore, vectors of 21, $21^2 = 441$ and $21^3 = 9261$ dimensions are required respectively for $k = 1, 2$, and 3 in this coding scheme. Each entry of the vector is associated with a possible permutation of k amino acids.

Since there is only a small collection of the possible 3-letter patterns appearing in each protein sequence, the dot product $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$ in the linear kernel for the tri-peptide composition calculates a value proportional to the number of tri-peptides coincide in two sequences $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The efficacy of the tri-peptide coding scheme in prediction of protein folds and subcellular localization is essentially due to the successful capture of local similarity by the coding scheme.

However, a more sensitive and biologically realistic coding method would allow some degree of mismatching in the tri-peptide representation. That is, the similarity should be large if the two sequences share many similar tri-peptides. This idea has been proposed and explored by Leslie *et al.* [13] for protein homology detection, and a set of spectral kernels was developed. In this work, the concept of mismatch kernel is explored in an implicit and different way. Here the discussion is restricted to case $k = 3$ for the simplicity of presentation, but the idea can be generalized to cases $k > 3$. In order to define the new kernel, we introduce a matrix in which each entry corresponds to the pairwise score of any two tri-peptides. For example, 12 for AAA-AAA pair, 11 for AAY-ACY pair, and 6 for TVW-TVR pair, if the BLOSUM62 matrix is used. The size of the matrix is 9261×9261, however, the matrix is only for the description and is never explicitly used in computation. Since majority of these pairs are associated with lower scores, the elimination of those pairs can reduce noise that may hinder the prediction. In addition, this also reduces training time. Accordingly, only a very small portion of the entries corresponding to high-scored pairs are kept, and other entries are replaced by 0 in the matrix. The matrix is called *the matrix of high-scored pairs of tri-peptides*, and is denoted as D. The new kernel $k(\cdot, \cdot)$ is defined as

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|D\boldsymbol{x}_i - D\boldsymbol{x}_j\|^2).$$

The term $\|D\boldsymbol{x}_i - D\boldsymbol{x}_j\|^2$ can be considered as a new sequence similarity for the two coding vectors of tri-peptide compositions. The similarity is measured between the transformed vectors $D\boldsymbol{x}_i$ and $D\boldsymbol{x}_j$, instead of the similarity between the original tri-peptide coding vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The example in Fig. 1 describes the coding vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ based on tri-peptide compositions and the transformed vectors $D\boldsymbol{x}_1$ and $D\boldsymbol{x}_2$ for the two short sequences of amino acids AAACY and ADCCY.

The selection of the high-scored pairs of tri-peptides is virtually filtering the tri-peptides sharing more than two residues in common. The concept of the mis-

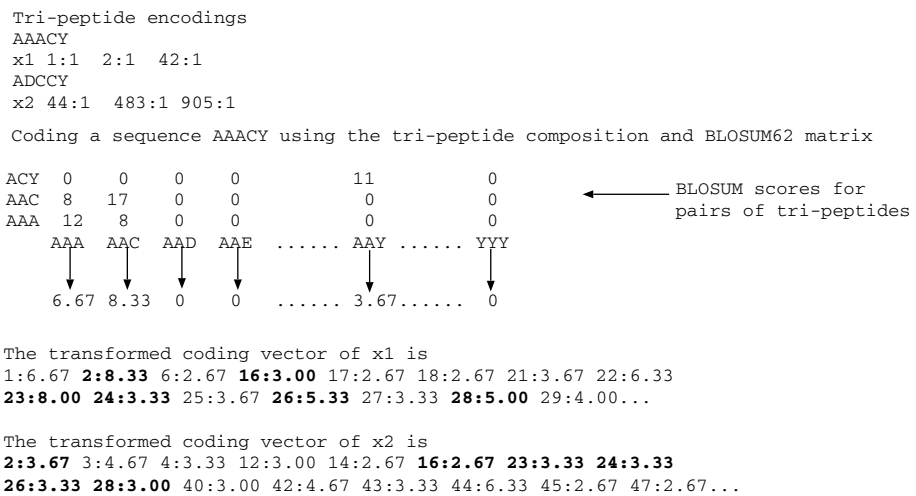


Fig. 1. The coding vectors for sequences AAACY and ADCCY based on the tri-peptide compositions and the transformed vectors based on the matrix of high-scored pairs of tri-peptides. The representation of coding vectors follows the sparse format of SVM-Light [12], i.e., the numbers appeared in the format of **vector index : score**. The shared elements between x_1 and x_2 are boldfaced

match string is explored, since only those mis-matched tri-peptides can yield high scores and survive the selection.

3 Experimental Results and Discussion

We employed the SVMs in conjunction with the coding vectors extracted by the method described above for training and testing. The evaluation of the methods was conducted on the following dataset.

Dataset. The set of proteins from Gram-negative bacteria used in the evaluation of PSORT-B [8] was considered (available at `http://www.psort.org/`) in this experiment. It consists of 1443 proteins with experimentally determined localizations. The dataset comprises 1302 proteins resident at a single localization site: 248 cytoplasmic, 268 inner membrane, 244 periplasmic, 352 outer membrane, and 190 extra cellular; it additionally contains a set of 141 proteins resident at multiple localization sites: 14 cytoplasmic/inner membrane, 50 inner membrane/periplasmic, and 77 outer membrane/extracellular. In our experiment, we considered only the 1302 proteins possessing a single localization.

Experiment and Results. We have compared the performance of the new kernel with that of the coding schemes based on the conventional k-peptide compositions using the above data set. The pairs of tri-peptides with scores

greater than a cut-off value 8 were selected to form the nonzero entries the matrix D of high-scored pairs of tri-peptides. This accounts for about 1.3% of the entries in matrix D. To ease the computational burden, the 2000 top scored entries from a transformed vector were further selected to form the input vector for SVMs. The cut-off value 8 and the number 2000 were determined empirically from the preliminary study to achieve the good predicting performance and fast training. The BLOSUM62 matrix was used for the assignment of scores to pairs of tri-peptides.

The experiment was carried out using a 5-fold cross-validation for each specific localization. Each time, the relevant dataset consisting of the proteins with the specific localizations was designated as the positive set; and the remainder of the proteins was denoted as the negative set. The radial basis function was chosen as the kernel function for the SVMs, since a preliminary experiment indicated that this kernel exhibited better performance.

As the sizes of the positive and negative sets are substantially different, the performance of SVMs was evaluated for precision (or sensitivity), defined as $tp/(tp + fp)$, and recall (or positive prediction value), defined as $tp/(tp + fn)$, where tp, tn, fp, and fn are the numbers of predicted true positive, true negative, false positive and false negative, respectively. The F-score combining the precision and recall was also provided: F-score=2*precision*recall/(precision+recall). The precision, recall, and F-score of the 5-fold cross-validation was computed respectively as the average of the values from 5 folds.

The generalization performance of an SVM is controlled by the following parameters:

(1) C : the trade-off between the training error and class separation;
(2) γ : the parameter in the radial basis function $\exp(-\gamma\|D\boldsymbol{x}_i - D\boldsymbol{x}_j\|^2)$;
(3) J : the biased penalty for errors from positive and negative training points.

The penalty term $C\sum_{i=1}^{m}\xi_i$ in SVM is splitted into two terms:

$$C\sum_{i=1}^{m}\xi_i \Rightarrow C\sum_{\{i:y_i=1\}}\xi_i + CJ\sum_{\{i:y_i=-1\}}\xi_i.$$

The choices of the parameters in this experiment are given as follows:
for the new kernel

C: from 1 to 40 with an incremental size of 3;
γ: from 0.001 to 1 with an incremental size of 0.003;
J: from 0.1 to 3.0 with an incremental size of 0.4;

and for the rest of the methods

C: from 1 to 150 with an incremental size of 10;
γ: from 1 to 100 with an incremental size of 10;
J: from 0.1 to 3.0 with an incremental size of 0.2.

The SVMLight package was used as the SVM solver [12]. The values of precision and recall of a 5-fold cross-validation are computed for each triplet (C, γ, J). The best values of precision, recall and the corresponding F-score for

Table 1. Results obtained from four different methods for the proteins from Gram-negative bacteria

Method	composition			di-peptide			tri-peptide			new method		
Localization	P	R	F	P	R	F	P	R	F	P	R	F
Cytoplasmic	80.09	70.77	74.66	81.12	57.69	66.09	83.43	45.00	55.09	77.38	73.48	75.38
Inner membrane	98.52	82.27	89.54	98.15	81.51	88.80	99.52	80.75	89.01	97.29	85.27	90.88
Periplasmic	94.12	55.17	68.38	91.80	54.14	65.77	90.37	50.34	63.11	85.98	68.45	76.22
Outer membrane	87.86	84.23	85.74	90.12	79.76	84.00	93.15	83.29	87.79	96.25	86.73	91.24
Extra cellular	88.38	53.68	66.05	89.71	53.68	66.27	92.57	50.53	64.63	92.11	64.86	76.12
Average	**89.79**	**69.23**	**76.87**	**90.18**	**65.36**	**74.18**	**93.17**	**64.80**	**74.62**	**89.80**	**75.76**	**81.97**

Composition, di-peptide and tri-peptide represents the method using the coding vector of the amino acid composition, di-peptide composition and tri-peptide composition, respectively. The symbols P, R and F stand respectively for precision, recall and F-score.

each method are given in Table 1. The new kernel based method demonstrated superior performance over the other three methods. The recall is improved substantially to a level of 75.76, from 69.23 (Composition), 65.36 (di-peptide), and 64.80 (tri-peptide).

The performance of the new kernel method also compares favorably with SCL-BLAST [16], a BLAST search based predictor for all localizations. The new method improves recall from 60.40 to 75.76 and F-score from 74.36 to 81.97, while having a lower precision (89.80) compared to that 96.70 of *SCL-BLAST*.

It is worth noting that the new method yields a similar overall performance comparing with PSORT-B, which gives precision 95.88, recall 73.20 and F-score 82.59. The PSORT-B comprises six modules designed for the prediction of specific localization sites. It is surprising that our single module can match the performance of this integrative predictor.

4 Conclusions

This work has introduced a novel kernel based on a matrix formed by the BLOSUM scores assigned to pairs of mis-matched tri-peptides of protein sequences. This kernel has been used in support vector machines for the prediction of subcellular localizations. The performance of the new kernel was empirically evaluated on a set of proteins with experimentally determined localizations from Gram-negative bacteria. Compared with the other coding systems using k-peptide compositions, the experimental results demonstrated that the new kernel exhibited superior overall performance for the prediction. The method also achieved a similar level of overall performance comparing with that of the integrated system PSORT-B.

References

1. Bannai, H., Tamada, Y., Maruyama, O., Nakai, K., Miyano, S.: Extensive feature detection of N-terminal protein sorting signals. Bioinformatics **18** (2002) 298–305

2. Cai, Y.D., Chou, K.C.: Predicting subcellular localization of proteins in a hybridization space. Bioinformatics **20** (2003) 1151–1156
3. Chou, K.C., Cai, Y.D.: Using Functional Domain Composition and Support Vector Machines for Prediction of Protein Subcellular Location. J. Biol. Chem. **277** (2002) 45765–4576
4. Cristianini, N., Shawe-Taylor, J.: *An Introduction to Support Vector Machines*, Cambridge University Press (2000)
5. Emanuelsson,O., Nielsen,H., Brunak,S., von Heijne, G.: Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. J. Mol. Biol. **300** (2000) 1005–1016
6. Emanuelsson, O.: Predicting protein subcellular localisation from amino acid sequence information. Brief. Bioinform. **3** (2002) 361–376
7. Feng, Z.P.: Prediction of the subcellular location of prokaryotic proteins based on a new representation of the amino acid composition. Biopolymers **58** (2001) 491–499
8. Gardy, J.L. *et al.*: PSORT-B: improving protein subcellular localization prediction for Gram-negative bacteria. Nucleic Acids Res. **31** (2003) 3613–3617.
9. von Heijne, G.: Signals for protein targeting into and across membranes. Subcell. Biochem. **22** (1994) 1–19
10. Horton, P., Nakai, K.: PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. Trends Biochem. Sci. **24** (1999) 34–36
11. Hua, S., Sun, Z.: Support vector machine approach for protein subcellular localization prediction. Bioinformatics **17** (2001) 721–728
12. Joachims, T.: Making Large Scale SVM Learning Practical. Advances in Kernel Methods-Support vector learning. MIT Press, Cambridge (1999)
13. Leslie, C., Eskin, E., Cohen, A., Weston, J., Noble, W.: Mismatch string kernels for discriminative protein classification. Bioinformatics **20** (2004) 467–476
14. Lu, Z., Szafron, D., Greiner, R., Lu, P., Wishart, D.S, Poulin, B., Anvik, J., Macdonell, C., Eisner, R.: Predicting subcellular localization of proteins using machine-learned classifiers. Bioinformatics **20** (2004) 547–556
15. Menne, K. M. L., Hermjakob, H., Apweiler, R.: A comparison of signal sequence prediction methods using a test set of signal peptides. Bioinformatics **16** (2000) 741–742
16. Nair, R., Rost, B.: Sequence conserved for subcellular localization. Protein Sci. **11** (2002) 2836–2847
17. Nakai,K. (2000) Protein sorting signals and prediction of subcellular localization. Adv. Protein. Chem., 54, 277-344
18. Nakai, K., Kanehisa, M.: Expert system for predicting protein localization sites in Gram-negative bacteria. Proteins, **11** (1991) 95–110
19. Nielsen, H., Engelbrecht, J., Brunak, S., von Heijne, G.: A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. Int. J. Neural Syst. **8** (1997) 581–599
20. Reinhardt, A., Hubbard, T.: Using neural networks for prediction of the subcellular location of proteins. Nucleic Acids Res. **26** (1998) 2230–2236
21. Tusnady, G.E., Simon, I.: Principles governing amino acid composition of integral membrane proteins: application to topology prediction. J. Mol. Biol. **283** (1998) 489–506
22. Tusnady, G.E., Simon, I.: The HMMTOP transmembrane topology prediction server. Bioinformatics **17** (2001) 849–850
23. Yu, C.S., Lin, C.J., Hwang, J.K.: Predicting subcellular localization of proteins for Gram-negative bacteria by support vector machines based on n-peptide compositions. Protein Sci. **13** (2004) 1402–1406

Reconstructing Phylogenetic Trees of Prokaryote Genomes by Randomly Sampling Oligopeptides

Osamu Maruyama[1], Akiko Matsuda[2], and Satoru Kuhara[3]

[1] Faculty of Mathematics, Kyushu University, Hakozaki 6-10-1, Higashi-ku, Fukuoka 812-8581, Japan
om@math.kyushu-u.ac.jp
[2] Graduate School of Systems Life Sciences, Kyushu University, Hakozaki 6-10-1, Higash0i-ku, Fukuoka 812-8581, Japan
3sl03012s@sls.kyushu-u.ac.jp
[3] Graduate School of Genetic Resources Technology, Kyushu University, Hakozaki 6-10-1, Higashi-ku, Fukuoka 812-8581, Japan
kuhara@grt.kyushu-u.ac.jp

Abstract. In this paper, we propose a method for reconstructing phylogenetic trees of a given set of prokaryote organisms by randomly sampling relatively small oligopeptides of a fixed length from their complete proteomes. For each of the organisms, a vector of frequencies of those sampled oligopeptides is generated and used as a building block in reconstructing phylogenetic trees. By this procedure, phylogenetic trees are generated independently, and a consensus tree of the resulting trees is obtained. We have applied our method to a set of 109 organisms, including 16 Archaea, 87 Bacteria, and 6 Eukarya, using less 10 % of all the 3,200,000 oligopeptides of length 5. Our consensus tree agrees with the tree of Bergey's Manual in most of the basic taxa. In addition, they have almost the same quality as the trees of the same organisms reconstructed using *all* the 20^K oligopeptides of length $K = 5$ and 6 given by Qi *et al.* Thus we can conclude that, the frequencies of a relatively small number of oligopeptides of length 5, even if those oligopeptides are determined in a random method, has phylogenetic information almost equivalent to the frequencies of *all* the oligopeptides of length 5 or 6.

1 Introduction

One of the interesting problems in computational biology is that of reconstructing phylogenetic trees of microbial organisms from their genomes and/or proteomes. Furthermore, more available genomes and proteomes are making this problem more challenging. A merit of studying this problem is that we can gain an insight into evolution. In most of the previous works on this problem, a distance-matrix between organisms is calculated and then a phylogenetic tree is generated from the matrix using a distance-based phylogenetic tree generator like UPGMA[14], Neighbor-Joining (NJ)[12], BIONJ[6], and NeighborNet[1]. Almost all the definitions of "distance" between two organisms are formulated

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 911–918, 2005.
© Springer-Verlag Berlin Heidelberg 2005

based on *sequence alignment*, including gene content [13, 15], presence-absence of families of genes [4, 8], presence-absence of genes in clusters of orthologous genes [16], supertree[2], and all-against-all genome-wide pairwise comparison [7].

Recently, Qi *et al.* [11] proposed a method for this problem, in which a vector of attributes based on frequencies of all the oligopeptides of a fixed length $K = 5$ or 6 in the complete proteome is used, in order to resolve the major branchings with Archaea and Bacteria more accurately than the results of the previous works and to avoid the inherent faults in sequence alignment. The distance between two arbitrary organisms is determined based on the angle between the corresponding vectors of the organisms. The matrix of those distances is calculated, and is taken as input to a phylogenetic tree generator, Neighbor-Joining [12]. They have applied their method to a set of 109 organisms, including 16 Archaea, 87 Bacteria, and 6 Eukarya, using a supercomputer, and reported that their phylogenetic trees constructed from the attribute vectors agree with the biologists' "tree of life" [5] in a majority of basic branchings, and especially, in all lower taxa.

Some natural questions of the work done by Qi *et al.* [11] are raised. *All* the frequencies of 20^K oligopeptides were used in their method. This requirement imposes a huge computation task, which is almost impossible in a current personal computer. For example, if $K = 6$, for each of the 109 organisms, the frequencies of 64,000,000 oligopeptides in the proteome should be calculated. Actually, a supercomputer was used to carry out their work. Our first question is whether the dimension of those vectors of oligopeptide frequencies can be reduced. Even if this is possible, how oligopeptides effective in reconstructing phylogenetic trees are selected? The second question is whether there are other sequence attributes of oligopeptides that are effective in reconstructing phylogenetic trees. As various formulations of species-species distance based on sequence alignment were discussed, various sequence attributes based on oligopeptide frequency should be considered and compared in order to identify better ones.

In this paper, to gain some insight into these questions, we have proposed a novel method for reconstructing phylogenetic trees using *random sampling* of oligopeptides. In our method, a relatively small number N of oligopeptides $r_1, r_2, \ldots, r_N$ of a fixed length $k - 1$ are randomly sampled from the complete proteomes of all the given organisms. Let $\Sigma = \{a_1, \ldots, a_{20}\}$ be an alphabet of symbols representing the 20 amino acids. For all the length-k oligopeptides $r_i a_j$ with $1 \leq i \leq N$ and $1 \leq j \leq 20$, the frequencies of the oligopeptides in the proteomes of an organism are calculated. The frequencies constitute an attribute vector of the organism. An additional advantage of this strategy is that we can afford to set k to be larger, e.g., $k = 12$, even if the computation is carried out on a personal computer. The attributes that we have applied in this work are the attribute of oligopeptide frequency used in [11] and another oligopeptide frequency attribute based on a k-th order Markov chain over a proteome. It should be noted here that the framework of our method is general such that various kinds of attributes based on oligopeptides are applicable. This random sampling procedure for making a phylogenetic tree is repeated several

times independently. A consensus tree of the resulting trees is calculated using CONSENSE [10], whose software is provided as a part of PHYLIP [3].

In our preliminary computational experiments, we have applied our method to the set of 109 organisms, including 16 Archaea, 87 Bacteria, and 6 Eukarya, that is the same set of species as [11]. Our resulting trees agree with the biologists' "tree of life" [5] in most of the basic taxa. In addition, they have almost the same quality as the trees of the same organisms reconstructed using all the 20^K oligopeptides of length $K = 5$ or 6, given by Qi *et al.* [11]. Thus we can conclude that, with respect to the attribute of oligopeptide frequency in [11], a relatively small number of oligopeptides of length 5 that are derived from randomly sampled oligopeptides of length 4 has information almost equivalent in reconstructing phylogenetic trees to *all* the oligopeptides of length 5 or 6.

2 Materials and Methods

In this section, the materials and methods we use here are described.

The set of organisms we use in this work is the same as the set of the organisms used in [11], which is 109 organisms, including 16 Archaea, 87 Bacteria, and 6 Eukarya, to compare our phylogenetic trees with their ones. The set of the protein sequences of an organisms is downloaded from "Genome Assembly/Annotation Projects" at the NCBI ftp site [17].

An overview of our method is given in Fig.1. Let n be the number of organisms. For $1 \leq i \leq n$, we denote the set of all the protein sequences of the i-th organism by S_i. Let $\mathcal{S} = \{S_1, \ldots, S_n\}$. For a positive integer k, the length of oligopeptides which are randomly samples from their complete proteomes is set to be $k-1$. Let N be the number of different oligopeptides which are randomly sampled. An *attribute function* α is a function $\alpha : \Sigma^\ell \times 2^{\Sigma^*} \to \mathcal{R}$, where $\Sigma = \{a_1, \ldots, a_{20}\}$ is an alphabet of symbols representing the 20 amino acids. We call the first argument of α, which is an oligopeptide of length ℓ, a *seed* of α. The second argument of α is supposed to be the set of all the protein sequences of an organism. The details of each step in Fig. 1 are described as follows.

Input: positive integers k, N, and R.
Output: a phylogenetic tree.
Procedure:

1. Repeat the followings R times independently.
   (a) A set of N different oligopeptides of length $k-1$ is randomly chosen from $\mathcal{S}$. The set is denoted by $P = \{p_1, \ldots, p_N\}$.
   (b) For $i = 1, \ldots, n$, a vector $u_i = (\alpha(p_1a_1, S_i) \cdots \alpha(p_1a_{20}, S_i) \cdots \alpha(p_Na_1, S_i) \cdots \alpha(p_Na_{20}, S_i))$ is calculated.
   (c) A distance matrix M among the n organisms is calculated from $u_1, \ldots, u_n$.
   (d) A phylogenetic trees is constructed from M.
2. A consensus tree of all the resulting trees is generated by CONSENSE[10].

Fig. 1. An overview of our method

The determination of seeds of an attribute function α is not completely random, but the prefixes of length $k-1$ of seeds are chosen randomly from the proteomes of the given organisms. For each randomly chosen prefix r, the strings ra of length k are used as seeds, where a is a symbol representing an amino acid. In this sense, the set of the resulting seeds is *biased*, which is more effective in reconstructing phylogenetic trees than a set of completely randomly sampled oligopeptides of length k (data not shown). The procedure of the determination of seeds is as follows. First, the following things are, in sequence, chosen uniformly at randomly, respectively: (i) a species i, (ii) a protein sequence s in S_i, (iii) a position $p \in \{1, \ldots, |s| - k + 2\}$. Then, the substring of s of length $k-1$ starting at p is added into a set P. This procedure is repeated until $|P| = N$. Next, from each sampled oligopeptide r, 20 different oligopeptides ra are generated, which are used as seeds of an attribute function. Thus, in this case, the parameter ℓ of α turns to be k. The size of the resulting set of seeds is $20N$, denoted by N'. Thus, the vector of α for the i-th organism is $(\alpha(p_1 a_1, S_i) \ldots \alpha(p_N a_{20}, S_i))$, whose dimension is N'.

Here, the attribute function of oligopeptide frequency used in [11], which we denote by α_1, is described briefly. Let $x = x_1 \cdots x_\ell$ be a string of length ℓ over Σ and $S \subseteq \Sigma^*$. We denote the number of occurrences of x in $s \in S$ by $c(x, s)$.

$$
\begin{aligned}
p(x, S) &= \sum_{s \in S} c(x, s) \Bigg/ \sum_{s \in S} (|s| - \ell + 1) \\
p^0(x, S) &= p(x_1 \cdots x_{\ell-1}, S) p(x_2 \cdots x_\ell, S) / p(x_2 \cdots x_{\ell-1}, S) \\
\alpha_1(x, S) &= \begin{cases} \frac{p(x,S) - p^0(x,S)}{p^0(x,S)} & \text{if } p^0(x, S) \neq 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

A feature of this attribute function is that the expectation $p^0(x, S)$ of $p(x, S)$ is subtracted from $p(x, S)$. This expectation is derived from a Markov model. Note that the attribute function is a natural generalization of "dinucleotide frequency" [9]. Another attribute function we consider in this work, denoted by α_2, is described as follows. This is designed based on a partial $(\ell - 1)$-th order Markov chain.

$$
\alpha_2(x, S) = \sum_{s \in S} c(x, s) \Bigg/ \sum_{s \in S} \sum_{a' \in \Sigma} c(x_1 \cdots x_{\ell-1} a', s) \ ,
$$

Thus, $\alpha_2(x, S)$ represents the probability that the amino acid x_ℓ occurs just after $x_1 \cdots x_{\ell-1}$ occurs in the sequences in S. A feature of this attribute is that the attribute is much easier to calculate rather than that of [11].

The distance between two arbitrary organisms is determined by the angle between the corresponding two attribute vectors. Let X and Y be attribute vectors corresponding to two organisms. The distance between X and Y is defined as $D(X, Y) = \frac{1 - C(X,Y)}{2}$ where $C(X, Y) = \sum_{k=1}^{N'} X_k Y_k \Big/ \sqrt{\sum_{k=1}^{N'} X_k^2 \cdot \sum_{k=1}^{N'} Y_k^2}$. Note that the lengths of the vectors are not considered in this definition.

A phylogenetic tree is generated from n attribute vectors of the n organisms using Neighbor-Joining [12], whose software is available as a part of a software package PHYLIP [3]. Note that an output tree of the method is unrooted. In the final step of our procedure, a consensus tree of the resulting trees calculated using R different sets of N oligopeptides of length k is generated by the CONSENSE[10] algorithm, whose software is also available as a part of PHYLIP[3].

The difference between our procedure with the attribute function α_1 and that of [11] is that in our procedure, a relative small N' oligopeptides are used as seeds of α_1, on the other hand, all the 20^K oligopeptides of length K are used in [11]. The remaining parts, including the definition of the distance between two organisms, the phylogenetic tree generator, and the consensus tree generator are the same as those of [11].

We define a measure to evaluate how a reconstructed phylogenetic tree agrees with the biologists' "tree of life" shown in [5]. Let T be a unrooted phylogenetic tree with the trifurcation of the three main domains, Archaea, Bacteria, and Eukarya. Thus this measure is defined for a tree with the trifurcation. The rooted tree derived from T by making the trifurcation node rooted is denoted by T'. Let E be the set of the branches in T'. For a branch $e \in E$, by L_e we denote the set of all the organisms assigned to the leaves that are descendants of the child node of e. For a set C of organisms, we can determine the unique edge e_C such that $C \subseteq L_{e_C}$ and $|L_{e_C}|$ is minimized. We call e_C the *separator* for C in T. The *unity* of C in T is defined as $u(T, C) = |C|/|L_{e_C}|$. The following taxa are considered to be C: *Archaea*(A), *Bacteria*(B), *Eukaryotes*(E), *Crenarchaeota*(A1), *Euryarchaeota*(A2), *Cyanobacteria*(B10), *Proteobacteria*(B12), *Alphaproteobacteria*(B12.1), *Betaproteobacteria*(B12.2), *Gammaproteobacteria*(B12.3), *Epsilonproteobacteria*(B12.5), *Firmicutes*(B13), *Actinobacteria*(B14), *Chlamydiae*(B16), and *Spirochaetes*(B17). Each taxon can be denoted by the code for it in Bergey's Manual if exists, which is given in parentheses. Since Eukaryotes are not included in Bergey's Manual, 'E' is assigned to it temporarily.

3 Results

Tab. 1 shows how the consensus phylogenetic trees constructed by the method in Fig. 1 with $k = 5$ and the trees in Fig. 1 and Fig. 2 in [11] agree with the biologists' "tree of life" shown in [5]. Recall that the tree in Fig. 2 and Fig. 1 in [11] are generated using *all* the oligopeptides of length 5 and 6, respectively, and that the trees agree with the biologists' "tree of life" [5] in a majority of basic branchings, and especially, in all lower taxa. As we can see in Tab. 1, the consensus trees constructed by the random sampling method with $k = 5$ have achieved almost the same quality as the tree in Fig. 2 and Fig. 1 in [11]. Surprisingly, the unities in the consensus tree with $N' = 280,000$ are all better or equal to those in Fig. 2 in [11]. It should be noted here that the number of seeds of α_1 used in constructing these trees are $N' = 280,000$ and $3,200,000$, respectively, so the former is only 8.75% of the latter. For Fig. 1 in [11], the

Table 1. A summary of the consensus trees constructed with the parameter sets (k, N', R) with $k = 5$, $R = 10$, and $N' = (16 + 4j) \times 10^4 \quad (0 \leq j \leq 7)$. Each column labeled with number N'', except the last two columns, corresponds to the consensus tree constructed with the parameter $N' = N'' \times 10^3$. Note that all those trees have the trifurcations (data not shown). The column labeled with "Qi2" is given for the consensus tree of Fig. 2 in [11], which is constructed by bootstrap with all the oligopeptides of length 5. The column labeled with "Qi1" is given for the tree of Fig. 1 in [11], which is constructed by a deterministic way with all the oligopeptides of length 6. The columns labeled with "u" shows the unities for the taxa shown in the first column, and the columns labeled with "f" shows the frequencies of appearances of the separators for the taxa, which are scaled between 0 and 1

taxon	160	200	240	280	320	360	400	440	Qi2	Qi1
	u, f	u, f	u, f	u, f	u, f	u, f	u, f	u, f	u, f	u
E	1.00, 0.5	1.00, 0.6	1.00, 0.7	1.00, 0.8	1.00, 0.9	1.00, 0.9	1.00, 0.9	1.00, 0.8	1.00, 0.95 ∼ 1.00	1.00
A	1.00, 0.9	1.00, 0.8	1.00, 0.8	1.00, 0.8	1.00, 0.9	1.00, 0.9	1.00, 0.9	1.00, 1.0	1.00, 0.95 ∼ 1.00	1.00
A1	1.00, 0.9	1.00, 0.7	1.00, 0.6	1.00, 0.8	1.00, 0.7	1.00, 1.0	1.00, 0.7	1.00, 0.9	1.00, 0.95 ∼ 1.00	1.00
A2	0.75, 0.9	0.75, 0.8	0.75, 0.8	0.75, 0.8	0.75, 0.9	0.75, 0.9	0.75, 0.9	0.75, 1.0	0.75, 0.95 ∼ 1.00	1.00
B	1.00, 0.8	1.00, 0.7	1.00, 0.8	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 0.9	1.00, 0.95 ∼ 1.00	1.00
B10	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 0.95 ∼ 1.00	1.00
B12	0.50, 0.2	0.59, 0.1	0.43, 0.8	0.49, 0.1	0.43, 1.0	1.00, 0.1	0.50, 0.4	0.60, 0.2	0.49, ∼ 0.85	1.00
B12.1	0.26, 0.2	0.26, 0.3	0.10, 0.8	1.00, 0.5	0.10, 1.0	0.26, 0.5	0.26, 0.7	1.00, 0.5	1.00, ∼ 0.85	0.26
B12.2	0.60, 0.5	1.00, 0.6	1.00, 0.7	0.60, 0.4	1.00, 0.5	0.60, 0.4	0.60, 0.5	0.60, 0.7	0.13, 0.85 ∼ 0.90	0.13
B12.3	0.88, 0.5	0.65, 0.3	0.25, 0.8	0.88, 0.6	0.25, 1.0	0.88, 0.4	0.65, 0.7	0.88, 0.7	0.88, 0.85 ∼ 0.90	0.64
B12.5	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 0.95 ∼ 1.00	1.00
B13	0.68, 0.1	0.31, 0.7	0.68, 0.2	0.31, 1.0	0.60, 0.3	0.36, 0.1	0.68, 0.2	0.31, 0.9	0.31, 0.95 ∼ 1.00	0.31
B14	1.00, 0.7	1.00, 0.9	1.00, 1.0	1.00, 0.9	1.00, 1.0	1.00, 0.9	1.00, 0.9	1.00, 0.8	1.00, 0.95 ∼ 1.00	1.00
B16	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 1.0	1.00, 0.95 ∼ 1.00	1.00
B17	0.03, 0.8	0.03, 0.7	0.03, 0.8	0.04, 0.2	0.03, 1.0	0.04, 0.1	0.03, 1.0	0.03, 0.9	0.03, 0.95 ∼ 1.00	0.03

consensus tree with $N' = 360,000$, given in Fig. 2, is quite a similar to the tree with respect to the unity scores. Our tree wins the unities of B12.2, B12.3, and B13 and loses only the unity of A2. A drawback of the consensus tree in Fig. 2 is that *E. coli* (B12.C3.O13.F1.G1) and *S. flexneri 2a* (B12.C3.O13.F1.G37) are mixed, although the other genera come together as they should.

These results imply that the set of all the oligopeptides of a fixed length are redundant with respect to reconstructing phylogenetic trees through the attribute function α_1. In addition, we can say that if a certain amount of oligopeptides are used to be seeds of α_1, it is not important how to select oligopeptides effective in reconstructing phylogenetic trees.

4 Discussion

For the attribute function α_1, we have also constructed phylogenetic trees with the different values of k, the length of oligopeptides sampled randomly. For $k = 6, \ldots, 15$, we have carried out our method with $N' = 100,000$, $200,000$ and $400,000$, and $R = 6$. Interestingly, all the consensus trees are inferior to those of $k = 5$ so some amount of oligopeptides of length 5 would have more phylogenetically valuable information on microbial organisms.

We have also applied our method with another attribute function α_2, which is an attribute based on a k-th order Markov chain over a proteome. We have

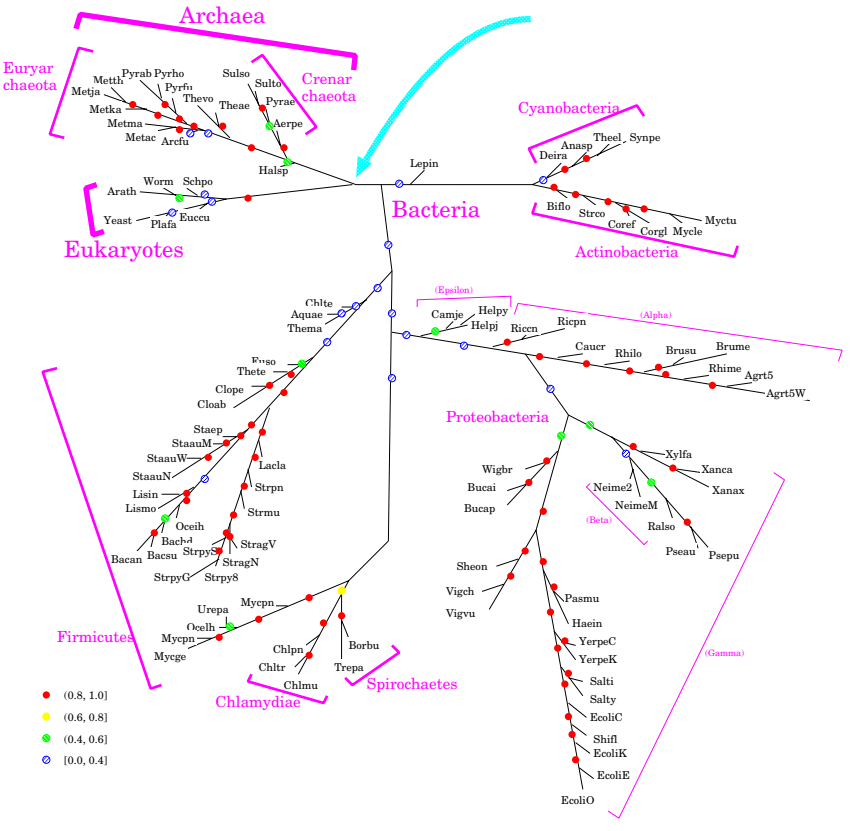


Fig. 2. A phylogenetic tree for 109 organisms constructed with the parameters $k = 5, R = 10$, and $N' = 360,000$. The abbreviations of the names of the 109 organisms are assigned to the corresponding leaves, respectively. The definitions of those abbreviations can be found in [11]. The trifurcation of the three domains is pointed out by the bold long arrow. Circles on the branches show the range of frequencies f of appearances of the branches: $f \in (0.8, 1.0]$ (red), $f \in (0.6, 0.8]$ (yellow), $f \in (0.4, 0.6]$ (green), and $f \in [0.0, 0.4]$ (blue)

constructed consensus trees with the same parameter sets as the cases of α_1. The result is that any consensus trees have no trifurcations. Further investigation into attribute functions would be needed.

Acknowledgments

This work was supported in part by Research for the Future Program of JSPS, and Grant-in-Aid for Scientific Research on Priority Areas (C) and Young Scientists (B) of MEXT.

References

1. D. Bryant and V. Moulton. Neighbor-net: an agglomerative method for the construction of phylogenetic networks. *Mol. Biol. Evol.*, 21:255–265, 2004.
2. V. Daubin, M. Gouy, and G. Perrière. A phylogenomic approach to bacterial phylogeny: evidence of a core of genes sharing a common history. *Genome Res.*, 12:1080–1090, 2002.
3. J. Felsenstein. PHYLIP (Phylogeny Inference Package) version 3.5c. Distributed by the author. Department of Genetics, University of Washington, Seattle, 1993. http://evolution.genetics.washington.edu/phylip.html.
4. S.T. Fitz-Gibbon and C.H. House. Whole genome-based phylogenetic analysis of free-living microorganisms. *Nucleic Acids Res.*, 27:4218–4222, 1999.
5. G.M. Garrity, J.A. Bell, and T.G. Lilburn. *Taxonomic Outline of the Procaryotes Release 5.0 Bergey's Manual of Systematic Bacteriology, 2nd edition.* Springer-Verlag, New York, May 2004.
6. O. Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.*, 14:685–695, 1997.
7. S.R. Henz, D.H. Huson, A.F. Auch, K. Nieselt-Struwe, and S.C. Schuster. Whole-genome prokaryotic phylogeny. *Bioinformatics*, 2004.
8. C.H. House and S.T. Fitz-Gibbon. Using homolog groups to create a whole-genomic tree of free-living organisms: an update. *J. Mol. Evol.*, 54:539–547, 2002.
9. S. Karlin and C. Burge. Dinucleotide relative abundance extremes: a genomic signature. *Trends Genet.*, 11:283–290, 1995.
10. T. Margush and F.R. McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 43:239–244, 1981.
11. J. Qi, B. Wang, and B.-I. Hao. Whole proteome prokaryote phylogeny without sequence alignment: a k-string composition approach. *J. Mol. Evol.*, 58:1–11, 2004.
12. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.
13. B. Snel, P. Bork, and M.A. Huynen. Genome phylogeny based on gene content. *Nat. Genet.*, 21:108–110, 1999.
14. R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.
15. F. Tekaia, A. Lazcano, and B. Dujon. The genomic tree as revealed from whole proteome comparisons. *Genome Res.*, 9:550–557, 1999.
16. Y.I. Wolf, I.B. Rogozin, N.V. Grishin, R.L. Tatusov, and E.V. Koonin. Genome trees constructed using five different approaches suggest new major bacterial clades. *BMC Evol. Biol.*, 1, 2001.
17. ftp://ftp.ncbi.nih.gov/genomes/.

Phylogenetic Networks, Trees, and Clusters

Luay Nakhleh[1] and Li-San Wang[2]

[1] Department of Computer Science, Rice University,
Houston, TX 77005, USA
nakhleh@cs.rice.edu
[2] Department of Biology, University of Pennsylvania,
Philadelphia, PA 19104, USA
lswang@mail.med.upenn.edu

Abstract. Phylogenetic networks model evolutionary histories in the presence of non-treelike events such as hybrid speciation and horizontal gene transfer. In spite of their widely acknowledged importance, very little is known about phylogenetic networks, which have so far been studied mostly for specific datasets.
Even when the evolutionary history of a set of species is non-treelike, individual genes in these species usually evolve in a treelike fashion. An important question, then, is whether a gene tree is "contained" inside a species network. This information is used to detect the presence of events such as horizontal gene transfer and hybrid speciation. Another question of interest for biologists is whether a group of taxa forms a clade based on a given phylogeny. This can be efficiently answered when the phylogeny is a tree simply by inspecting the edges of the tree, whereas no efficient solution currently exists for the problem when the phylogeny is a network. In this paper, we give polynomial-time algorithms for answering the above two questions.

1 Introduction

Phylogenies are the main tool for representing the relationships among biological entities. Their pervasiveness has led biologists, mathematicians, and computer scientists to design a variety of methods for their reconstruction. Furthermore, extensive studies have been focused on the performance of these methods under different models and settings, as well as on the combinatorial and biological properties of trees (e.g., [7, 2]). However, almost all such methods construct trees, and almost all studies have been aimed at trees. Yet, biologists have long recognized that trees oversimplify our view of evolution, since they cannot take into account such events as hybridization, lateral gene transfer, and recombination. These non-tree events give rise to edges that connect nodes on different branches of a tree, giving rise to a directed acyclic graph structure that is usually called a *phylogenetic network*.

A gene tree is a model of how a gene evolves through duplication, loss, and nucleotide substitution. Gene trees can differ from one another as well as from the species phylogeny. Such differences arise during the evolutionary process due to events such as duplication and loss, whereby each genome may end up with multiple copies of a given gene—but not necessarily the same copies that survive in another genome. Unless the

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 919–926, 2005.
© Springer-Verlag Berlin Heidelberg 2005

genome is very well sampled, only a subset (sometimes only one copy, in fact) of the gene is used in phylogenetic analyses. As a result, the phylogeny for the gene may not agree with the species phylogeny, nor with the phylogeny for another gene. Because the gene copy has a single ancestral copy, barring recombination, the resulting history is a branching tree. Point mutations can cause some of the copies to be imperfect representations of the original, but this process does not compromise the existence of the (gene) tree. Events such as recombination, hybrid speciation, and lateral gene transfer break up the genomic history into many small pieces, each of which has a strictly treelike pattern of descent [4]. Thus, within a species phylogeny, many tangled gene trees can be found, one for each nonrecombined locus in the genome. Incongruence among gene trees is a powerful tool for detecting recombination, hybrid speciation, and other non-treelike evolutionary events (e.g., see [6]). While testing for incongruence between two (gene) trees can be done in a straightforward manner, it is not as simple for testing the incongruence between a tree and a network, since the number of trees "inside" a network grows exponentially with the number of non-treelike events. In this paper, we give the first polynomial-time algorithm for solving this problem.

A phylogeny can be viewed as a collection of clusters of taxa (each defined as the set of leaves in a subtree). Various approaches for reconstructing phylogenies (trees and networks) have been proposed based on this view (see, e.g., [1, 3]). An interesting biological question, then, is whether a group of taxa forms a cluster in a given phylogeny. This question can be answered in a straightforward manner when the phylogeny is a tree, since each edge in a tree defines a unique cluster. However, the number of clusters in a phylogenetic network grows exponentially with the number of non-treelike events, and hence an efficient algorithm for solving the problem is not straightforward. In this paper, we present the first polynomial-time algorithm for solving this problem.

2 Phylogenetic Trees and Networks

A *rooted* phylogenetic tree is a rooted tree without redundant nodes (nodes of indegree and outdegree 1) and whose leaves are labelled distinctively (by a set of taxa). An unrooted phylogenetic tree is a rooted phylogenetic tree with the root suppressed. Every edge e in an unrooted leaf-labeled tree T defines a bipartition (or, split) $\pi(e)$ on the leaves (induced by the deletion of e), so that we can define the set $\Pi(T) = \{\pi(e): e \in E(T)\}$. Every edge e in a rooted leaf-labeled tree T defines a cluster $c(e)$ of leaves (those leaves that are reachable from the root through e), so that we can define the set $C(T) = \{c(e): e \in E(T)\}$. A *clade* of a rooted tree T is the entire subtree rooted at a node of T; the set of all leaves in a clade correspond to a *cluster* of T.

Evolutionary events such as hybrid speciation and lateral gene transfer result in networks of relationships. In hybrid speciation, two lineages recombine to create a new species, as symbolized in Fig. 1(a). Prior to hybridization, each site on each homolog has evolved in a tree-like fashion, although, due to meiotic recombination, different strings of sites may have different histories. Thus, each site in the homologs of the parents of the hybrid evolved in a tree-like fashion on one of the trees contained inside (or induced by) the network representing the hybridization event, as illustrated in Fig. 1(b) and Fig. 1(c).

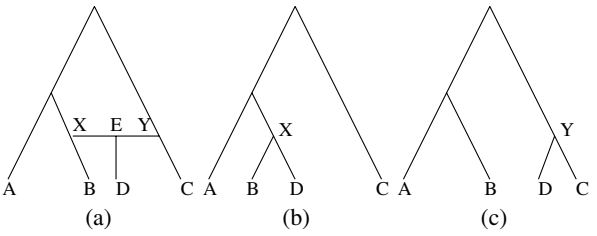


Fig. 1. A network with a single hybrid speciation event, and its two induced trees

2.1 Phylogenetic Networks: Model and Properties

We adopt the general model of (reduced) phylogenetic networks given in [5].

Definition 1. *A* **phylogenetic network** *is a connected directed acyclic graph* $N = (V, E)$*, where* V *can be partitioned into* $\{r\} \cup Tr(N) \cup Nt(N) \cup L(N)$*, where:*

1. *Node* r *is the root; it has indegree* 0*.*
2. *Set* $Tr(N)$ *is the set of tree nodes; each node* u *in* $Tr(N)$ *has indegree* 1 *and outdegree* > 1*.*
3. *Set* $Nt(N)$ *is the set of network nodes; each node* v *in* $Nt(N)$ *has indegree* 2 *and outdegree* 1*.*
4. *Set* $L(N)$ *is the set of leaf nodes (taxa); each node* x *in* $L(N)$ *has indegree* 1 *and outdegree* 0*. Each node* x *in* $L(N)$ *is labeled uniquely by an integer* i*, where* $1 \leq i \leq |L(N)|$*.*

Figure 1(a) shows an example of a phylogenetic network. Given a network N, we classify its edges as *tree edges* and *network edges*. An edge $e = (u, v)$ is a tree edge if v is a tree node or a leaf; otherwise, it is a network edge. We say that network N is *binary* if the root and all tree nodes of N have outdegree 2. In this paper, and unless noted otherwise, all networks are binary. Further, we assume that if u is a tree node and (u, v) and (u, w) are the two edges incident from u, then at least one of the two nodes v and w is a tree node.

A *forced contraction* is an operation on a graph in which we delete a redundant node and replace the two edges incident to it by a single edge. We generalize the *clade* concept to networks as follows. Given a network N, we say that the DAG N', rooted at node x, is a *network clade* of N, if there exists an edge $e = (u, x)$ in N whose removal disconnects N, thus creating two components, one of which is N' (rooted at x). If network clade N' does not contain network nodes, i.e., N' is a tree, we refer to N' simply as a clade. Given a network N, and a clade N', we say that N' is *maximal* if N does not contain any clade N'' such that $N' \subset N''$.

If there is a directed path from node u to node v ($u \neq v$) in a directed tree T, we say that u is *above* v and that v is *below* u, both denoted by $u > v$. We say that node u in N is a *lowest network node* if (1) u is a network node, and (2) for any network node v, $v \neq u$, we have $u \not> v$. We write $N|_{L'}$, where $L' \subset L(N)$, to denote the subgraph N' obtained from N by removing all leaves not in L', and then applying forced contraction operations and removal of nodes of outdegree 0 (other than the leaves in L').

2.2 The Relationship Between Networks and Trees

There is a fundamental biological relationship between (species) networks and (gene) trees, as described above. We now formalize this concept mathematically.

Let N be a network with p network nodes $h_1, h_2, \ldots, h_p$. Further, assume that the two edges incident into h_i are e_{i_1} and e_{i_2}. An *inheritance profile*, IP, for N is a set of size p which contains exactly one of the two edges e_{i_1} and e_{i_2} for each network nodes h_i. A rooted tree T is *induced* by (or, contained in) a network N if there exists an inheritance profile IP such that T can be obtained from N as follows: for network node h_i, if $e_{i_1} \in IP$, remove edge e_{i_2}; otherwise, remove edge e_{i_1} (and then apply forced contraction operations to the resultant graph). Biologically, the evolutionary history of a gene within the species network corresponds to a tree T induced by N. Associated with this tree is an inheritance profile IP that determines how to obtain T from N; in this case, we say that IP is a *valid* inheritance profile that induces T. We denote by $\mathcal{T}(N)$ the set of all trees induced by network N. A network N induces (or, contains) a cluster C, $C \subseteq L(N)$, if there exists a tree T such that N induces T and C is a cluster of T. We denote by $\mathcal{C}(N)$ the set of all clusters induced by network N.

As mentioned before, deciding whether a cluster or a tree are induced by a given network plays a significant role in solving major biological problems such as network reconstruction, gene tree and species network relationships. We now formalize the two decision problems.

Problem 1. (THE NETWORK-TREE CONTAINMENT PROBLEM)

Input: A phylogenetic network N and a tree T.
Question: Does N contain T?

Problem 2. (THE NETWORK-CLUSTER CONTAINMENT PROBLEM)

Input: A phylogenetic network N and a cluster C.
Question: Does N contain C?

A trivial approach for solving the Network-Cluster Containment Problem is to find "the" lowest common ancestor, x, of C in the network N, and test whether the cluster is contained in the network clade rooted at x. This approach may fail for (at least) two reasons: (1) x may not be unique in a network, and (2) the network clade rooted at x may contain many of the network nodes of N, in which case the search for a solution would take time that is exponential in the number of network nodes, and hence, probably the network size. In Section 5 we give polynomial time algorithms for these two problems. In order to obtain these results, we first introduce the concept of *network decomposition* which forms the basis for our algorithms.

3 Network Decomposition

Before we give the technical details of our algorithms, we describe the network representation we use, which is vital for achieving the running times of the algorithms in the next sections. We assume that a network N is represented using an $n \times n$ adjacency

matrix M^N, where n is the number of nodes in the network. We have $M^N[u,v] = 1$ if there is an edge $(u,v) \in E(N)$, and $M^N[u,v] = 0$ otherwise. Using this representation, a forced contraction operation takes $O(1)$ time, and an edge deletion takes $O(1)$ time, as well.

3.1 Preprocessing Networks

An *SH-loop* (speciation-hybridization) is a cycle that contains only network edges, and that consists of two paths p_1 and p_2, such that p_1 and p_2 starting from the same tree node v_0, pass through two sets of network nodes, and end at the same network node v_1. Let $e_1 = (v_0, x)$ and $e_2 = (v_0, y)$ be the two network edges incident from v_0. We break the SH-loop by removing either e_1 or e_2, and applying forced contraction operations to all redundant nodes. We repeat the same process until N is SH-loop-free, i.e., N does not contain any SH-loops.

Theorem 1. *A network $N = (V, E)$ can be preprocessed in $O(|E(N)|^2)$ time.*

Proposition 1. *Let N be a phylogenetic network, and let N' be the network obtained after the preprocessing. Then, $\mathcal{T}(N) = \mathcal{T}(N')$.*

3.2 Maximal Clades and Connections

Unless noted otherwise, all networks are SH-loop free. Given a phylogenetic network N, we seek to decompose N into maximal-size clades and disjoint subgraphs of N that connect those clades. To formalize this, we first define some concepts.

Given a node x in network N, we say that a network node y ($y \neq x$) in N is *x-convergent* if any directed path from y to a leaf of N passes through x. Given a maximal clade A of N, and the root a of A, we say that subgraph J of N is the *connection* of A if J is the subgraph obtained by restricting N to all a-convergent nodes and their incident edges.

Lemma 1. *Let A and J be a clade and its connection, respectively, in a network N. Then, when reversing the orientation of its edges, J has a rooted tree topology, where each leaf is a tree node in N and each internal node is a network node in N. Further, the root of J is a lowest network node.*

Definition 2. *A* T-decomposition *of a network N is an ordered set of pairs $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$, where A_i and J_i are a maximal clade and its connection, respectively, in N_i (N_i is obtained by removing the subgraphs A_{i-1} and J_{i-1} from N_{i-1}, except for the leaves of J_{i-1}, i.e., the tree nodes, and applying forced contraction operations to the resultant graph; for the base case, $N_1 = N$); m is the cardinality of the decomposition.*

Theorem 2. *A T-decomposition of a network N is computable in $O(|Nt(N)||V(N)|)$ time.*

Let (u, v) be a terminal edge (i.e., and edge incident with a leaf) that belongs to connection J_i; v is a tree node in N. If N is binary, then for the three edges incident to v, two belong to the same component, because v is suppressed in the i^{th} step in the

decomposition algorithm. We define $\iota(u, v)$ to be the index of this component. It is straightforward to show that $\iota(u, v) > i$. Finally, we show that exactly one terminal edge from each component in a T-decomposition is used to induce a tree T.

Lemma 2. *If T is a tree induced by a network N, and D is a T-decomposition of N, then exactly one terminal edge from each connection in D is used to induce that tree.*

3.3 Dependency Graphs

Given a network N and its T-decomposition D, we define the *dependency digraph* $K^{N,D}$ as follows.

Definition 3. *Given a network N and its T-decomposition $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$, the dependency graph is a directed multigraph $K^{N,D}$, where node v_i in $K^{N,D}$ corresponds to the pair (A_i, J_i) in D, and edge (v_i, v_j) $(i > j)$ in $K^{N,D}$ corresponds to a terminal edge connecting J_j and J_i in N.*

In other words, $K^{N,D}$ is the graph resulting from replacing each component (A_i, J_i) in D by a single node v_i, and hence, $K^{N,D}$ is necessarily connected. If $K^{N,D}$ had a cycle, then N would be cyclic. Therefore, we have the following result.

Proposition 2. *The dependency graph $K^{N,D}$ is connected and acyclic, and is computable in $O(|Nt(N)||V(N)|)$ time. Moreover, (v_i, v_j) is an edge in $K^{N,D}$ only if $i > j$.*

4 Reduced Inheritance Profiles and the Cluster Lemma

Given a T-decomposition D of cardinality m, a *reduced inheritance profile* is a set of size m that contains exactly one terminal edge per connection in the decomposition. We only keep the terminal edges because all inheritance profiles having the same set of terminal edges necessarily induce the same tree. A reduced inheritance profile extends into an inheritance profile in a straightforward manner, as no edges in the reduced inheritance profile are incident with the same network node. We say that a reduced inheritance profile is *valid* if it induces a tree. The following results show the correspondence between inheritance profiles and reduced inheritance profiles.

Proposition 3. *Let D be a T-decomposition of a network N. Then,*

1. *For each valid reduced inheritance profile IP there exists a valid inheritance profile IP' that contains IP and induces the same tree.*
2. *For each valid inheritance profile IP' there exists a unique valid reduced inheritance profile IP that induces the same tree.*

The dependency graph can be seen as a compact representation, mainly for reduced inheritance profiles.

Lemma 3. *Let D be a T-decomposition of a network N, $K^{N,D}$ be the dependency graph, and IP be a valid reduced inheritance profile. Then, $K^{N,D}$, restricted to the edges in IP, forms a tree.*

Algorithm TestCinN(N,C)

1. Compute a T-decomposition $D = ((A_1, J_1), \ldots, (A_m, J_m = \emptyset))$.
2. Test if C can be decomposed into the following form: $\bigcup_{i \in \psi(C)} L(A_i) \cup L'$, where $L' = \emptyset$ or $L' \subset L(A_l)$ for some l. If not, return NO. If $L' = \emptyset$ then let $l = \max_{i \in \psi(C)} i$.
3. Partition $V = V(K^{N,D})$ into two sets: $V_C = \{v_i | i \in \psi(C)\}$ and $\overline{V_C} = V - V_C$. Compute the set $E_C = \{e_{ij} = (v_i, v_j) | e_{ij} \in E(K^{N,D}), v_i \in \overline{V_C}, v_j \in V_C, j \neq l\}$.
4. If $L' \neq \emptyset$, test if L' is a cluster of A_l. If not, return NO; otherwise:
   (a) Let v' be the root of the clade whose leaf set is L'.
   (b) For each terminal edge (u, v) in A_i, for some $i \in \psi(C)$ and $\iota(u, v) = l$, (edge (u, v) connects the i'th component to the l'th component), add (u, v) to E_C if u is not a descendant of v' in N_i.
5. Remove all terminal edges in N and that correspond to edges in E_C (and apply forced contraction operations); let the result be N_C. If N_C is connected, return YES. Otherwise, return NO.

Fig. 2. Algorithm **TestCinN** for deciding the NETWORK-CLUSTER CONTAINMENT PROBLEM

We are now in position to show the correlation between clusters and a T-decomposition of a network – a result that forms the basis for our algorithms.

Lemma 4. (Cluster Lemma) *Let $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$ be a T-decomposition of a network N. Each cluster C induced by N can be written as $C = \cup_j C_j$, where each C_j is an element of $\{L(A_i) : 1 \leq i \leq m\}$, except for at most one of the C_j's, which may be a proper subset of an element of $\{L(A_i) : 1 \leq i \leq m\}$.*

Corollary 1. *Let $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$ be a T-decomposition of a network N, IP be a reduced inheritance profile, and C be a cluster. Then, when restricted to the nodes whose corresponding maximal clades have nonempty intersection with C and to the edges in IP, the dependency graph $K^{N,D}$ forms a tree. Further, the root of that tree has the highest index.*

Corollary 1 gives an algorithm for computing the component that contains the node which "determines" a cluster C.

5 Polynomial-Time Algorithms for the Decision Problems

5.1 Deciding the Network-Cluster Containment Problem

We are finally in a position to describe a polynomial-time algorithm for deciding the Network-Cluster Containment Problem. The algorithm is given in Fig. 2. Let $D = \{A_i, J_i\}_{1 \leq i \leq m}$ be a T-decomposition of a network N, and let $C \subset L(N)$ be a cluster. We define the set $\psi(C) = \{i : 1 \leq i \leq m \text{ and } L(A_i) \subseteq C\}$. The basic idea is to compute a set E_C of edges that are *incompatible* with C, i.e., edges that cannot co-exist with C in the same tree induced by N.

Theorem 3. *Algorithm TestCinN(N,C) decides the Network-Cluster Containment Problem in $O(|V(N)|^2)$ time.*

Algorithm TestTinN(N,T)

1. Compute a T-decomposition $D = ((A_1, J_1), \ldots, (A_m, J_m = \emptyset))$.
2. For each nontrivial cluster C in T ($C \neq L(N)$ and $|C| > 1$), call TestCinN(N,C); update N by removing E_C from N.
3. If N is connected, return YES; otherwise, return NO.

Fig. 3. Algorithm **TestTinN** for deciding the NETWORK-TREE CONTAINMENT PROBLEM

Corollary 2. *Given any network N, a phylogenetic tree T, and a cluster C in T, N induces T if and only if N_C induces T.*

5.2 Deciding the Network-Tree Containment Problem

Using algorithm TestCinN(N,C), Fig. 3 describes our polynomial-time algorithm for deciding the Network-Tree Containment Problem.

Theorem 4. *Algorithm TestTinN(N,T) decides the Network-Tree Containment Problem in $O(|V(N)||L(N)|)$ time.*

References

[1] D. Bryant and V. Moulton. NeighborNet: An agglomerative method for the construction of planar phylogenetic networks. In R. Guigo and D. Gusfield, editors, *Proc. 2nd Workshop Algorithms in Bioinformatics (WABI'02)*, volume 2452 of *Lecture Notes in Computer Science*, pages 375–391. Springer Verlag, 2002.

[2] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Inc., Sunderland, MA, 2003.

[3] D.H. Huson. SplitsTree: A program for analyzing and visualizing evolutionary data. *Bioinformatics*, 14(1):68–73, 1998.

[4] W.P. Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997.

[5] B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):13–23, 2004.

[6] L. Nakhleh, T. Warnow, and C.R. Linder. Reconstructing reticulate evolution in species – theory and practice. In *Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB 2004)*, pages 337–346, 2004.

[7] D.L. Swofford, G.J. Olsen, P.J. Waddell, and D.M. Hillis. Phylogenetic inference. In D.M. Hillis, B.K. Mable, and C. Moritz, editors, *Molecular Systematics*, pages 407–514. Sinauer Assoc., Sunderland, Mass., 1996.

SWAT: A New Spliced Alignment Tool Tailored for Handling More Sequencing Errors

Yifeng Li[1] and Hesham H. Ali[2]

[1] Department of Pathology and Microbiology,
University of Nebraska Medical Center, Omaha, NE 68198-6805
yl1@unmc.edu
[2] Department of Computer Science, College of Information Science and Technology,
University of Nebraska at Omaha, Omaha, NE 68182-0116
hali@mail.unomaha.edu

Abstract. There are several computer programs that align mRNA with its genomic counterpart to determine exon boundaries. Though most of these programs perform such alignment efficiently and accurately, they can only tolerate a relatively small number of sequencing errors. These programs also highly depend on the GT/AG rule in finding splice sites. Both properties make them less desirable in the case of aligning EST reconstructed transcript with genomic DNA to identify splicing variants, where a lot of sequencing errors and non-canonical splice sites are expected. Using a novel heuristic algorithm, we developed a tool that can handle much more sequencing errors. Test dataset results indicated that SWAT (Sequencing-error Well-handled Alignment Tool) has a much stronger error-handling ability than Sim4 and Spidey, two other popular spliced alignment tools. In the presence of up to 10 percent randomly introduced sequencing errors, it can still give the precise number of exons and exon boundaries in most cases. The robustness of SWAT makes it a desirable tool in cases where sequencing error is a concern. A web service is freely available at http://app1.unmc.edu/swat/swat.html.

1 Introduction

This work is motivated by recent studies on splicing variants using EST data [1, 2]. Alternative splicing is known as the most important mechanism to increase protein diversity. It is estimated that 40–60% of human genes undergo such event and altered mRNA splicing can have a dramatic effect on the structure of the encoded protein [3-5]. Due to its prevalence and biological importance, extensive studies on alternative splicing have been carried out. To identify different splice forms, it is essential to perform an alignment of the transcribed mRNA sequences with the corresponding genomic DNA. ESTs (expressed sequenced tags), which are derived from processed mRNA, in public database provide an ideal resource for transcripts that can be used in such alignment [6]. The major advantage of using ESTs is their abundance, which makes the database likely contain all differently spliced transcripts of the same gene. A disadvantage is their low quality. ESTs often contain errors at a rate much higher

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 927 – 935, 2005.
© Springer-Verlag Berlin Heidelberg 2005

than those of the finished or even draft genomic sequence [6, 7]. Though most existing tools perform the alignment efficiently and accurately, they usually only allow a relatively small number of sequencing errors. Therefore, to efficiently use ESTs, there is a call for more robust alignment tool. SWAT, which uses a novel heuristic algorithm, achieves fast alignment and efficient error-handling at the same time.

2 Previous Work

There are several programs that perform mRNA-to-genomic alignment. Among them, est_genome [8], Sim4 [9] and Spidey [10] are the three most widely known, whereas MGAlign [11] is the most recently developed one. The following is a brief review of the above-mentioned programs.

Est_genome uses a liner-space dynamic programming recurrence to align spliced sequences to their genomic counterparts. A modified Smith-Waterman scan is performed to locate the maximum-scoring segments. The genome sequence is then searched against forward and reverse strands of the spliced sequence, assuming the splice consensus GT/AG. Est_genome has a strong error-handling ability. However, the long running time limits its use.

Sim4 uses a heuristic algorithm. It first determines all the high-scoring segment pairs (HSPs) such as those computed by the BLAST program [12]. It then selects a best chain of the HSPs subject to certain constraints and finally applies the GT/AG rule to find exon boundaries. The program only expects a small number of sequencing errors. BLAST-based software suffers from the high granularity of the BLAST program and the problem is worsened when there are sequencing errors.

Spidey is a heuristic also. It first uses the BLAST to align mRNA and the genomic sequence. The BLAST alignments are then sorted by score and assigned into windows by a recursive function which takes the first alignment and then goes down the list to find all alignments that are consistent with the first. Once the genomic windows are constructed, another BLAST search is performed to align entire mRNA with each window at a lower stringency. Spidey then uses a greedy algorithm to generate a high scoring, non-overlapping subset of the alignments from the second BLAST search. Finally, alignments are truncated or extended as necessary so that they terminate at the splice donor site and do not overlap. Due to its BLAST-based manner, Spidey is not expected to handle sequencing errors efficiently either.

MGAlign is a newly developed tool which uses a rapid heuristic method. Its authors claim that it is more accurate and faster than Sim4 and Spidey [13]. However, the details of the algorithm have never been published. We found that this tool does have an improved ability in handling small exons but it is even more sensitive to sequencing errors than Sim4 and Spidey.

3 Proposed Approach

The original problem to be solved is to determine the exon-intron structure for a given pair of mRNA and its parent genomic DNA. SWAT uses a divided and conquer algo-

rithm: suppose an mRNA segment is found matching a DNA segment and such a match can not be extended further in either direction, then by removing the matched segments in both sequences, the original problem can be turned down into two identical but smaller ones (Figure 1). The segment removed is an exon whose boundary has been determined. The smaller cases resulted from the splitting can be treated exactly in the same way: finding the match and then splitting. This strategy can be used repeatedly until there is no mRNA segment left, at which point the original problem is solved. Therefore, the problem of determining exon boundaries can be solved recursively. After each cycle, it always results in an exon with solved boundaries and one or two identical but miniature version of the original problem. The base case for the recursive function will be as the following: given an mRNA and its parent genomic DNA, from a randomly selected point in the mRNA, finding the corresponding exon in both sequences that covers that point.

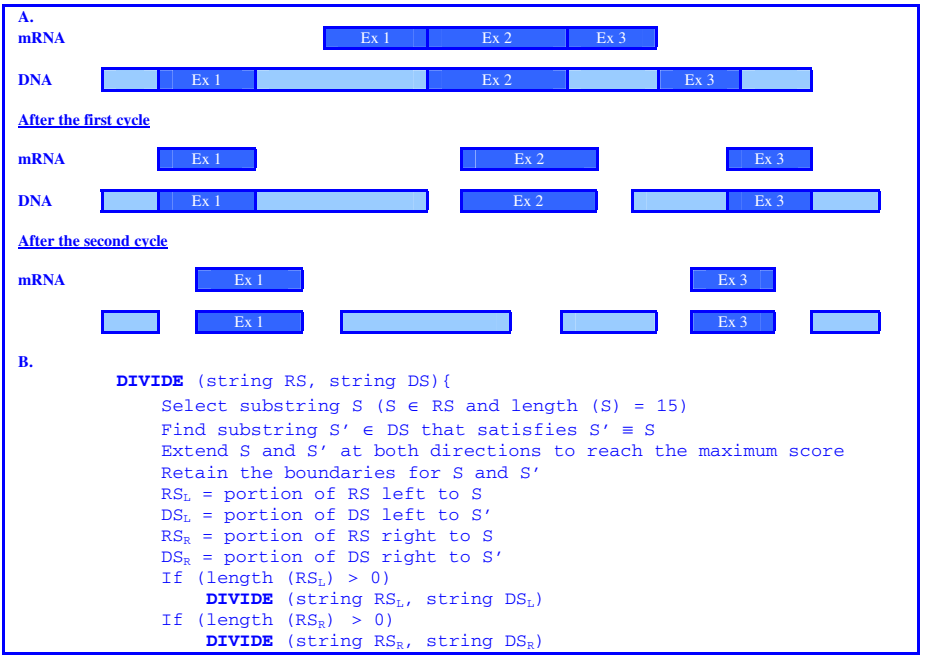


Fig. 1. Essence of the proposed divided and conquer algorithm. A. Schematic diagram of the key concept. B. Pseudo code for the recursive function call

In each cycle, the starting point in the mRNA is randomly chosen. Given the same pair of sequences the program may go a different path each time it runs, yet reaches the same result. Such randomness is essential in getting the best result when there are errors in the sequence. Current settings guarantee that wrong match will not happen when there are no or very little sequencing errors. However, the chance of having a

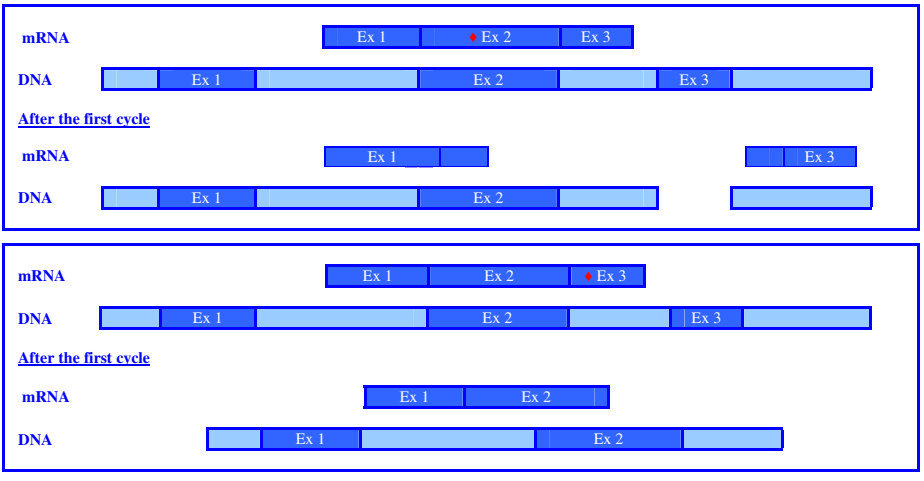


Fig. 2. Graphic illustration of the importance of random start. Top: Wrong match happens between part of Ex2 in the mRNA and a DNA region containing Ex3. This wrong match and subsequent splitting cause problem for finding matches in the next cycle. Bottom: Wrong match is remedied by selecting a different starting point in the mRNA. ♦: the starting point

wrong match enhances with increased sequencing errors. Mismatch dues to sequencing errors in a particular run cannot be avoided but the failure it causes can be remedied by running the program several times more. Suppose a wrong match happens in the middle of a run and results in a bad split which in turn causes problems for finding matches in the subsequent cycles, the overall similarity score would be much lower than what is expected. Since the starting point is randomly chosen, another run may go a different path and therefore avoid the bad match (Figure 2). Best result can always be reached by choosing the highest similarity score from 2 to 3 runs. The strategy of random start makes a contribution to the stronger error-handling ability of this tool.

4 Algorithm Description

The proposed algorithm is implemented using C++ programming language and the major steps are described below.

1. *Find the matched segments*. Randomly select an mRNA segment with 10-15 residues in length and find its exact match in the DNA. Extend the matches at both directions as long as exact match can still be found. When exact match is no longer available at either end, extend the current matches to 8 and 12 residues for mRNA and DNA, respectively. For all combinations of length in both extended segments, use the Needleman-Wunsch algorithm [14] to find the segment pair that gives the highest similarity score. This high-score pair will be used as the actual extension. Repeat this if each time it lets the matches grow more than one residue. Otherwise, stop there and no further attempt in extending the match is made. This extension strategy allows

certain number of mismatches, insertion or deletion to be tolerated in determining the final matched segments.

2. *Split both sequences and call the recursive function*. The matched segments detected in the previous step are removed from both sequences and the corresponding position of the cleavage sites are recorded for future exon-intron structure construction. By doing so, both sequences are split into a left portion and a right portion. Each portion of the mRNA and DNA pair forms an identical but miniature version of the original problem, which is solved through a recursive function call. The program steps out of the recursive function when there is no mRNA segment left.

3. *Trim the exon boundaries*. After determining all the candidate exons, GT/AG rule is applied to trim the exon boundaries.

5 Alignment Validation

SWAT was tested with the new multi_exon entries of the GENIE gene finding data set♠. There are totally 137 records in this data set. However, matches cannot be found for 5 of them in the GenBank records, so we actually tested 132 cases, of which 64 have 2-5 exons, 38 have 6-10 exons, 19 have 11-15 exons, 5 have 16-20 exons, 2 have 21-25 exons, 2 have 26-30 exons, 1 has 33 exons and 1 has 51 exons. In this study, GenBank records of each mRNA are aligned with the corresponding genomic DNA using our program and the results are compared with the NCBI annotation. For all 132 entries, results given by SWAT are consistent with the NCBI annotation. Like

Intron-Exon Structure Based on the Alignment

● SWAT

Your cDNA and genomic sequence has 3778 and 9674 bps, respectively.

	cDNA	GENOMIC	LENGTH	IDENTITY
EXON 1	1 111	1 111	111	100%
EXON 2	112 216	682 786	105	100%
EXON 3	217 1562	2856 4201	1346	100%
EXON 4	1563 1650	4390 4477	88	100%
EXON 5	1651 1805	4774 4928	155	100%
EXON 6	1806 1991	5111 5296	186	100%
EXON 7	1992 2120	5557 5685	129	100%
EXON 8	2121 2303	5816 5998	183	100%
EXON 9	2304 2400	6716 6812	97	100%
EXON 10	2401 2524	7569 7692	124	100%
EXON 11	2525 2699	7837 8011	175	100%
EXON 12	2700 2870	8226 8396	171	100%
EXON 13	2871 2980	8504 8613	110	100%
EXON 14	2981 3083	8809 8911	103	100%
EXON 15	3084 3749	9009 9674	666	100%

Fig. 3. Sample output of SWAT showing intron-exon structure based on the alignment

♠ The website for the data set is at http://www.fruitfly.org/seq_tools/datasets/Human.

MGAlign, SWAT also has a strong ability in detecting small exons, which tend to be missed by Sim4 and Spidey. Exon as small as 9 bps can be successfully detected. Two of such cases are included in the 'Alignment Examples' of the service website. In both cases, a small exon was missed by Sim4 and Spidey. A sample output is shown below (Figure 3). In this case, N-deacetylase/N-sulfotransferase 2 (NDST2) transcript is aligned with its corresponding genomic DNA.

6 Error-Handling Ability

For certain cases in the test dataset, the mRNA segment and the corresponding exon do not perfectly match due to sequencing errors. But for all such cases, the small discrepancy can be properly handled by the program. Two of such cases are included in the 'Alignment Examples' of the service website. Since SWAT is designed to handle more sequencing errors, its actual error-handling ability has been further tested with error-containing mRNA and the results are compared with those given by Sim4 and Spidey . For each of the 132 entries that are used previously for alignment validation, 10 percent sequencing errors were randomly introduced into the mRNA sequence. These error-containing mRNA were aligned with error-free DNA sequence using the three programs. In all cases, SWAT correctly determined the number of exons and in only 9 cases there were small shift in the exon boundaries[*]. The rate of failure for Sim4 and Spidey are much higher (Table 1). These results clearly show that SWAT has a stronger error handling ability than the other two programs.

Table 1. Result of error-handling ability comparison among SWAT, Sim4 and Spidey. The numbers shown are the failed cases

	SWAT	Sim4	Spidey
Incorrect boundaries only	9	34	57
Incorrect numbers & boundaries	0	15	32
Overall rate of failure	6.8%	37.1%	67.4%

A particular case is shown below to give a better idea of how robust the program is (Figure 4). The same example in Figure 3 is used here for easy comparison but now the mRNA sequence contains 10 percent randomly introduced errors. In this case, SWAT gives identical result as there were no errors. Only the identity drops which indicates there are indeed errors in the sequence. Sim4 gives a big shift for the left boundary of exon 1 possibly due to the higher than average local error rate. Spidey detects an extra exon. Result given by MGAlign was even worse where the alignment ends up with a lot of gaps and dramatic boundary shifts (result not shown).

[*] Shift within 3 residues is not considered as a failure and this standard is applied for all three programs.

30-111	30-111	86%
112-215	682-786	86%
216-1562	2856-4201	89%
1563-1650	4390-4477	88%
1651-1805	4774-4928	90%
1806-1991	5111-5296	94%
1992-2120	5557-5685	89%
2121-2303	5816-5998	88%
2304-2400	6716-6812	88%
2401-2524	7569-7692	88%
2525-2699	7837-8011	93%
2700-2870	8226-8396	90%
2871-2980	8504-8613	95%
2981-3083	8809-8911	93%
3084-3749	9009-9674	90%

	mRNA	genomic	identity
Exon 1	1-111	1-111	85.6%
Exon 2	112-211	682-781	87.0%
Exon 3	212-217	2193-2198	100.0%
Exon 4	218-1562	2857-4201	89.4%
Exon 5	1563-1650	4390-4477	88.6%
Exon 6	1651-1810	4774-4933	88.8%
Exon 7	1811-1991	5116-5296	95.0%
Exon 8	1992-2120	5557-5685	89.1%
Exon 9	2121-2287	5816-5982	89.8%
Exon 10	2301-2400	6713-6812	89.0%
Exon 11	2401-2524	7569-7692	87.9%
Exon 12	2525-2699	7837-8011	93.1%
Exon 13	2700-2870	8226-8396	90.1%
Exon 14	2871-2980	8504-8613	95.5%
Exon 15	2981-3083	8809-8911	93.2%
Exon 16	3084-3749	9009-9674	90.2%

Intron-Exon Structure Based on the Alignment

SWAT

Your cDNA and genomic sequence has 3778 and 9674 bps, respectively.

	cDNA	GENOMIC	LENGTH	IDENTITY
EXON 1	1 111	1 111	111	85.6%
EXON 2	112 216	682 786	105	86.7%
EXON 3	217 1562	2857 4201	1346	89.4%
EXON 4	1563 1650	4390 4477	88	88.6%
EXON 5	1651 1805	4774 4928	155	91%
EXON 6	1806 1991	5111 5296	186	94.6%
EXON 7	1992 2120	5557 5685	129	89.1%
EXON 8	2121 2303	5816 5998	183	88.5%
EXON 9	2304 2400	6716 6812	97	88.7%
EXON 10	2401 2524	7569 7692	124	87.9%
EXON 11	2525 2699	7837 8011	175	93.1%
EXON 12	2700 2870	8226 8396	171	90.7%
EXON 13	2871 2980	8504 8613	110	95.5%
EXON 14	2981 3083	8809 8911	103	93.2%
EXON 15	3084 3749	9009 9674	666	90.2%

Fig. 4. Error-handling ability comparison among Sim4, Spidey and SWAT on a test case. 10% sequencing errors are randomly introduced to the mRNA. Upper left: Result given by Sim4. Upper right: Result given by Spidey. Bottom: Result given by SWAT

7 Discussion

SWAT uses a divided and conquer algorithm. The original problem is kept turning down into identical but smaller ones and solved recursively. As indicated by the test dataset results, in most cases up to 10 percent randomly introduced sequencing errors can be properly handled. The strong error-handling ability first comes from the strategy used to extend local optimal match and then is further strengthened by the fact that best result can always be reached by choosing the highest similarity score from several runs, which is made possible by the divided-and-conquer algorithm itself and

the randomness in selecting the starting point. Besides its ability in handling errors, SWAT has several other desirable properties as well. First, the program is extremely fast. The average running time per alignment is less than 0.5 second when performed on a 1.40 GHz Intel Pentium 4 system with 256 Mb of RAM. Second, non-canonical sites are likely to be detected. Though GT/AG rule is applied in the program, similarity is given the first priority and under no circumstances will it be sacrificed to satisfy the rule. Third, due to the manner of how the original sequence is broken down, all subsequent exons found are in consistent order. Therefore, consistency (mRNA and genomic DNA are non-overlapping and linearly consistent) is automatically achieved and no post-alignment allocation is needed. Forth, best result is guaranteed. The random way in selecting the starting point in each cycle makes it always possible to choose the best result from several runs. Both its strong error handling ability and high speed make SWAT an ideal tool for mRNA and genomic sequence alignment especially when sequencing error is a concern.

Acknowledgements

We thank Dr. Mark Pauley, Xutao Deng and Alexandre Tchourbanov for their helpful comments and suggestions. We thank Yan Wu for running the test dataset. We would also like to thank Joe Ziskovsky for helping us set up the web service.

References

1. Kan Z, Rouchka EC, Gish WR and States DJ. Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. Genome Res. (2001) 11, 889–900
2. Modrek B, Resch A, Grasso C and Lee C. Genome-wide analysis of alternative splicing using human expressed sequence data. Nucleic Acids Res. (2001) 29, 2850–2859
3. Graveley BR. Alternative splicing: Increasing diversity in the proteomic world. Trends Genet (2001) 17: 100–107
4. Black DL. Protein diversity from alternative splicing: A challenge for bio-informatics and post-genome biology. Cell (2000) 103: 367–370
5. Modrek B and Lee C. A genomic view of alternative splicing. Nat Genet (2002) 30: 13–19
6. Jongeneel CV. Searching the expressed sequence tag (EST) database: panning for genes. Brief Bioinform (2000) 1: 76-92
7. Boguski MS, Lowe TM and Tolstoshev CM. dbEST — database for "expressed sequence tags". Nat Genet (1993) 4, 332-333
8. Mott R. EST_GENOME: A program to align spliced DNA sequences to unspliced genomic DNA. Comput. Appl. Biosci. (1997) 13**:** 477–478.
9. Wheelan SJ, Church DM and Ostell JM. Spidey: a tool for mRNA-to-genomic alignments. Genome Res. (2001) 11: 1952-1957
10. Florea L, Hartzell G, Zhang Z, Rubin GM and Miller W. A computer program for aligning a cDNA sequence with a genomic sequence. Genome Res. (1998) 8: 967-74
11. Lee BT, Tan TW and Ranganathan S. MGAlignIt: a web service for the alignment of mRNA/EST and genomic sequences. Nucleic Acids Res. (2003) 31: 3533-3536

12. Altschul SF, Gish W, Miller W, Myers EW and Lipman DJ. Basic Local Alignment Search Tool. J. Mol. Biol. (1990) 215, 403-410
13. Ranganathan S, Lee BT and Tan TW. MGAlign, a reduced search space approach to the alignment of mRNA sequences to genomic sequences. Genome Informatics (2003) 14: 474-475
14. Needleman, SB. and Wunsch, CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. (1970) 48: 443-453

Simultaneous Alignment and Structure Prediction of RNAs

Are Three Input Sequences Better Than Two?

Beeta Masoumi and Marcel Turcotte

School of Information Technology and Engineering,
University of Ottawa,
Ottawa, Ontario, Canada
{bmasoumi, turcotte}@site.uottawa.ca

Abstract. Comparative RNA sequence analyses have contributed remarkably accurate predictions. The recent determination of the 30S and 50S ribosomal subunits brought more supporting evidence. Several inference tools are combining free-energy minimisation and comparative analysis to improve the quality of secondary structure predictions. Using many input sequences should improve the accuracy, reduce the likelihood that bad predictions are made, but also lower the sensitivity. To investigate these claims, we have extended the software system Dynalign to use three input sequences, rather than two, and tested our algorithm with 10 tRNAs and 13 5S rRNAs. The following hypotheses were tested: 1) the use of three input sequences improves the average accuracy compared to predictions based on two input sequences. Also, it should be less likely that all three input sequences simultaneously fold into a bad free-energy minimum compared to predictions based on two sequences, consequently, 2) the worse prediction (minimum accuracy) for any sequence should be more accurate when three input sequences are used rather than two. Finally, the consensus structure of three sequences is probably less representative of the individual sequences. 3) Therefore, the average coverage should be less.

1 Introduction

The repertoire of known non-protein coding RNAs (ncRNAs) is growing rapidly[1]. The housekeeping roles of RNAs, such as those of the tRNA, rRNA, RNAseP, snRNA and snoRNA, were established early. In the recent years, it has become clear that RNAs also have important regulatory functions. Examples include microRNAs, which regulate the expression of protein genes by targeting a complementary region of their mRNAs. MicroRNAs constitute one of the most abundant class of regulatory molecules, and are key to many developmental processes[2]. Several discoveries collectively demonstrate that untranslated messenger RNAs can sense the level of metabolites, and modulate the expression of certain genes accordingly. Those RNAs are referred to as RNA sensors and riboswitches[3, 4]. Post-transcriptional regulation of gene expression often involves

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 936–943, 2005.
© Springer-Verlag Berlin Heidelberg 2005

secondary structure elements located in the untranslated regions of mRNAs[5]. Consequently, detailed knowledge of RNA secondary and tertiary structure is essential to help understand RNA functions.

RNA secondary structure prediction methods have been thoroughly evaluated. In particular, Gardner and Giegerich have performed a comprehensive evaluation of comparative RNA structure prediction methods[6]. Doshi et al. reviewed specifically free-energy minimisation methods that are using the nearest-neighbour model[7]. One of their main conclusions is that free-energy minimisation methods based on the nearest-neighbour model work best for shorter sequences, such as tRNA or 5S rRNA, for which they reported an average accuracy for predictions of 69% and 71% respectively.

Recently, Mathews and Turner developed, and published, a software system combining free-energy minimisation and comparative sequence analysis for finding the minimum free-energy structure common to two input sequences[8]. The computer system, called Dynalign, greatly improves the accuracy of secondary structure predictions compared to free-energy minimisation alone.

Herein, we extend this algorithm to use three input sequences, rather than two, and investigate the performance of the new computer program. We called this software system eXtended-Dynalign, or X-Dynalign for short, to emphasise its origin.

This paper is organised as follows. Section 2 outlines the algorithm. In Section 3, the main hypotheses to be empirically evaluated are laid out, the datasets are described, and the evaluation measures are defined. Section 4 presents the results. Section 5 concludes and discusses the results.

2 Algorithm

Dynalign is a pragmatic implementation of the algorithm proposed by Sankoff for solving simultaneously the RNA folding and alignment problems[9]. Dynalign is restricted to two input sequences, while the original proposal was formulated for an arbitrary set of N input sequences. Also, Dynalign introduces a constraint on the maximum distance between aligned nucleotides so as to reduce the execution time. This is analogous to the banding technique that is used for sequence alignment.

X-Dynalign is a direct extension of Dynalign. It takes as input three sequences and produces a three-way sequence alignment as well as a common secondary structure. The objective function consists of a linear combination of the free-energy of each sequence, given the common secondary structure, and an empirical term for gap penalties.

$$\Delta G^{\circ}_{\text{total}} = \Delta G^{\circ}_{\text{sequence 1}} + \Delta G^{\circ}_{\text{sequence 2}} + \Delta G^{\circ}_{\text{sequence 3}} + \Delta G^{\circ}_{\text{gaps}}$$

where $\Delta G^{\circ}_{\text{sequence } i}$, for $i \in \{1, 2, 3\}$, represents the conformational free-energy of the sequence i when folded onto the common secondary structure, according to the nearest-neighbour model.

Three sets of recurrence equations are defining the objective function: W, V and $W9$. Equations of the form $W(i, j, k, l, m, n)$ represent the minimum free-energy for the optimal alignment and structure prediction of $S_1[i..j]$, $S_2[k..l]$ and $S_3[m..n]$, when i, k and m are aligned, and j, l and n are also aligned, where S_i denotes the sequence i and $S_i[a..b]$ represents the fragment of S_i comprising the nucleotides $a, a + 1 \ldots b$. Equations of the form $V(i, j, k, l, m, n)$ represent the minimum free-energy assuming that i and j, k and l, and m and n are simultaneously aligned but also base-paired. Finally, $W9(i, k, m)$ represents the minimum free-energy for the prefix alignment of $S_1[1..i], S_2[1..k]$ and $S_3[1..m]$. A detailed description of the recurrence equations can be found in [10]. The recurrence equations are solved using dynamic programming. The algorithm requires $\mathcal{O}(|S_1|^2 M^4)$ space and $\mathcal{O}(|S_1|^3 M^6)$ time, where M is a constant that limits the maximum distance between aligned nucleotides.

3 Methodology

3.1 Experiments

The following hypotheses are tested: 1) the use of three input sequences should improve the average accuracy compared to predictions based on two input sequences. When three input sequences are used, the likelihood that they all three fold into a bad free-energy minimum should be less than when two input sequences are used, consequently, 2) the worse prediction (minimum accuracy) should be more accurate when three input sequences are used rather than two. Finally, the secondary structure common to three input sequences should be less representative of the individual sequences, consequently, 3) the average coverage should be less. But first, we determine empirically the optimum gap penalties for these datasets.

3.2 Datasets

Input sequences were selected such that they can be aligned optimally with a small value of M. Obviously, this information would not be known in advance in most cases. Also, the input sequences were filtered so that the maximum pairwise identity was less than 90%. A total of 10 tRNA sequences from the original paper were used. Their pairwise sequence identity varies from 27.3 to 68.8 %. The secondary structure assignments were taken from the compilation by Sprinzl et al.[11, 12]. A set of 13 5S rRNA sequences was built using information obtained from the Comparative RNA Web Site[13, 14, 15]. Their pairwise sequence identity varies from 47.2 to 88.2% .

3.3 Performance Measures

We call **references**, the secondary structures that were obtained from the tRNA compilation by Sprinzl and the Comparative RNA Web Site. We define as **true**

positives (TP) the base pairs that are occurring in both structures, reference and predicted, **false positives** (FP), the base pairs that are occurring in the predicted structure but not in the reference one, and **false negatives** (FN), the base pairs that are occurring in the reference structure but not in the predicted one. Offsets were not allowed.

The **positive predictive value** (sometimes called PPV, specificity or accuracy) is defined as the fraction of the predicted base pairs that are also present in the reference structure, $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$. The **sensitivity** (coverage) is defined as the fraction of the base pairs from the reference structure that are correctly predicted, $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$. Finally, we also measured the **Matthews Correlation Coefficient**, as defined by Gorodkin, Stricklin and Stormo[16]:

$$\sqrt{\frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FN})} \times \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FP})}}$$

4 Results

4.1 Calibrating Gap Penalties

In [8], the optimal gap penalty was found to depend on the class of RNA; 2.0 and 0.4 Kcal/mol for the tRNA and 5S rRNA, respectively. Accordingly, we performed two sets of experiments to measure the effect of various gap penalty scores on PPV, sensitivity and MCC. Since these experiments are time consuming, only six gap penalty scores were tested, $0.0, 0.25, 0.5, 1.0, 2.0, 4.0$, and only triples that can be aligned with a small value of M, here 5, were selected. In all, 105 and 90 predictions were made for the tRNA and 5S rRNA, respectively. For the experiments presented herein, we have chosen a gap penalty score of 1.0 Kcal/mol, because it corresponds to the maximum sensitivity for both datasets, tRNA and 5S rRNA.

4.2 Comparative Analysis

We present the analysis of the tRNA data first. Nine runs, 27 predictions, were made using X-Dynalign, while 19 runs, 38 predictions, were made using Dynalign. The mean PPV, sensitivity and MCC are 96.8 ± 7.6, 94.4 ± 7.5 and 95.6 ± 7.3 for X-Dynalign, and 92.1 ± 14.6, 89.1 ± 15.7 and 90.5 ± 15.0 for Dynalign. Our data represent a subset of that of Mathews and Turner, the PPV for Dynalign measured on this subset is 5.7 percentage points higher than theirs. We observe that the use of three sequences improves all three indices and reduces their variance, for this particular dataset.

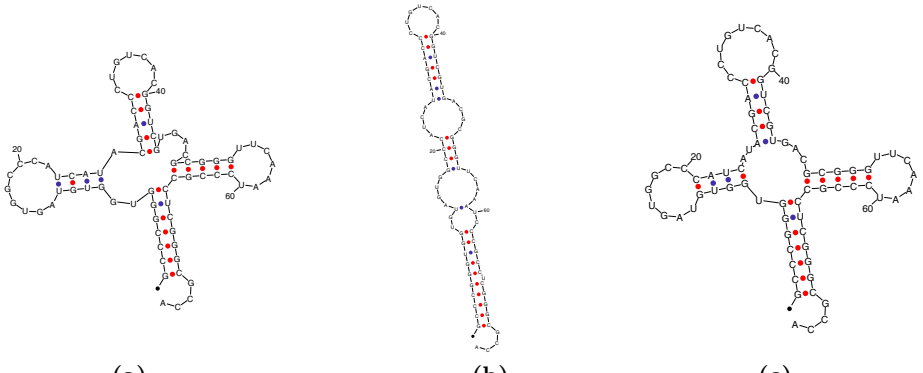
Table 1 presents the performance indices per sequence. Dynalign performed well in the best case scenario. For all the sequences, it was possible to find a pair of input sequences having a high positive predictive value. The maximum PPV for every entry is 100, except for that of RS0380. Further analysis shows that the structure of RS0380 ($\mathrm{tRNA}^{\mathrm{Asp}}$ *Haloferax volcanii*) has an extra stem

Table 1. PPV for the tRNA dataset. The subscripts xd and d are designating X-Dynalign and Dynalign respectively. N is the number of predictions

Id	N_{xd}	N_d	Min_{xd}	Min_d	Max_{xd}	Max_d	Ave_{xd}	Ave_d
RD0260	4	5	100	80	100	100	100.0	96.0
RD0500	4	5	76	45	100	100	82.2	80.8
RD4800	5	5	100	80	100	100	100.0	96.0
RE2140	2	4	100	100	100	100	100.0	100.0
RE6781	2	4	100	77	100	100	100.0	94.3
RF6320	4	5	95	45	100	100	96.4	89.1
RL0503	1	2	100	100	100	100	100.0	100.0
RL1141	2	3	100	70	100	100	100.0	90.3
RS0380	1	2	100	83	100	87	100.0	85.2
RS1141	2	3	100	70	100	100	100.0	90.3

in the variable loop, that X-Dynalign predicted more accurately. For 9 out of 10 experiments, the maximum sensitivity for X-Dynalign equals or exceeds that of Dynalign.

Both algorithms are seeking to find a structure that minimises a linear combination of the free-energy of each input sequence given the common structure. Using three input sequences should have a positive impact on the worse case scenario. It should be less likely that all three input sequences jointly fold into the wrong minimum free-energy structure than with two input sequences. Our data support this observation, for all the entries the minimum PPV for X-Dynalign is the same or better than that of Dynalign. For 8 out of 10 sequences, the minimum PPV is 100, in one case, the minimum PPV is 95, and for one case the minimum PPV is 76. The two sequences leading to the worse predictions are RD0500 and RF6320, see Figure 1. Dynalign produces an elongated structure.



(a) **(b)** **(c)**

Fig. 1. Reference (a), Dynalign (b) and X-Dynalign (c) secondary structures for the tRNA RD0500

Table 2. PPV for the 5S dataset

Id	N_{xd}	N_d	Min_{xd}	Min_d	Max_{xd}	Max_d	Ave_{xd}	Ave_d
AJ131594	2	3	100	91	100	100	100.0	94.5
AJ251080	6	5	88	82	90	86	90.3	84.8
D11460	6	5	87	66	87	88	87.6	79.4
K02682	8	9	63	88	100	97	89.1	92.0
M10816	3	4	90	85	90	88	90.7	87.8
M16532	1	2	94	77	94	85	94.1	81.8
M25591	6	5	87	82	90	86	89.8	84.8
V00336	3	4	75	65	100	100	91.9	91.4
X02024	9	6	88	82	90	88	90.1	85.8
X02627	1	2	100	92	100	100	100.0	96.0
X04585	2	3	72	68	94	93	83.4	82.7
X08000	5	5	90	88	90	90	90.6	89.4
X08002	5	5	90	88	90	90	90.6	89.4

However, using a third sequence increases the accuracy by more than 30 percentage points. The structure produced by X-Dynalign has the overall cloverleaf shape, however, the nucleotides of the first part and second part of the D-arm are shifted by one and two positions, respectively. The minimum coverage is generally good. For all the sequences the coverage is 75% or better. For all the tests the coverage for X-Dynalign is the same as Dynalign or better.

For our second test set, we have 19 runs, 57 predictions, using X-Dynalign, and 29 runs, 58 predictions, using Dynalign. The mean PPV, sensitivity and MCC are 90.3±5.8, 76.6±5.3 and 83.2±5.5 for X-Dynalign, and 87.7±7.4, 79.2± 6.7 and 83.3 ± 6.7 for Dynalign. For this particular dataset, the performance of both systems is comparable on the basis of the Matthews correlation coefficient. What is gained in accuracy is lost in sensitivity.

Table 2 presents the performance indices per sequence. Using three input sequences improves the worse (PPV) prediction for 12 out of 13 sequences. Also, for 10 out of 13 sequences, the minimum PPV obtained is 85% or more. The minimum sensitivity is the same or improved for 11 out of 13 sequences. However, the maximum sensitivity exceeds that of Dynalign for 2 out of 13 sequences.

The prediction of the 5S rRNA of *Micrococcus lutus* (K02682) has an accuracy of 63% only. We believe this is due to the fact that single base pair insertion has not been implemented yet in X-Dynalign. In the triple K02682, V00336 and X04585, the structure of *Rhodobacter capsulatus* (X04585) has a shorter helix IV, 7 base pairs compared to 8 for the other two structures.

5 Conclusion and Discussion

We have extended the software system Dynalign to use three input sequences, rather than two. The resulting system is called eXtended-Dynalign (X-Dynalign for short). Its time/space complexity limits its application to 1) short sequences (say less than

200 nt) and 2) sequences that can be aligned optimally with a small value of M (less than 6), where M is the maximum distance of aligned positions.

The strengths of Dynalign carry over to the new system. Namely, it improves the accuracy of secondary structure predictions compared to predictions based on a single input sequence. It requires no sequence homology.

It also shares some of its limitations. In particular, the gap penalties are treated as a separate term in the objective function. The optimal value has to be determined empirically. In [8], it was found that the optimal value for this term depends on the class of RNA studied. In our limited experiments, the dependency seems less important. It also seems that there is large plateau were several gap penalty scores are leading to a nearly optimal solution; w.r.t. PPV, for example. Our key conclusions are:

- The lowest PPV for any prediction is generally improved when using three input sequences;
- The average accuracy is improved;
- The average sensitivity of the algorithm slightly degraded for the 5S rRNA dataset. However, a per sequence analysis shows that the majority of the lowest sensitivity scores are higher for X-Dyanlign than Dynalign;
- X-Dynalign is able to reproduce subtle details, such as the prediction of a stem in the variable region of certain tRNAs.

There are several obvious directions for extending this class of algorithms, such as handling pseudo-knots and reporting suboptimal structures. However, one of the most urgent improvement is to reduce the time/space complexity. Several runs presented herein take up to week to compute on some of the fastest processors available today.

The detailed knowledge of RNA secondary structure is essential for understanding the sequence-structure-function relationships. X-Dynalign takes advantage of the paramount of data that is accumulating in sequence databases. Because it requires no sequence homology, X-Dynalign should be useful to comparative RNA sequence analyses.

6 Availability

The source code, written in C++, as well as the scripts for calculating the performance indices are made available under the GNU General Public Licence from `http://bio.site.uottawa.ca/software/x-dynalign`. Supplementary material, including additional tables and figures, can be found on our web site.

References

1. Storz, G.: An expanding universe of noncoding RNAs. Science **296** (2002) 1260–1263
2. Bartel, D.P.: MicroRNAs: Genomics, biogenesis, mechanism, and function. Cell **116** (2004) 281–297

3. Nudler, E., Mironov, A.X.: The riboswitch control of bacterial metabolism. Trends Biol. Sci. **29** (2004) 11–17
4. Lai, E.C.: RNA sensors and riboswitches: Self-regulating messages. Current Biology **13** (2003) R285–R291
5. Mignoe, F., Gissi, C., Liuni, S., Pesole, G.: Untranslated regions of mRNAs. Genome Biology **3** (2003) 0004.1–0004.10
6. Gardner, P.P., Robert, G.: A comprehensive comparison of comparative RNA structure prediction approaches. BMC Bioinformatics **5** (2004) 140
7. Doshi, K.J., Cannone, J.J., Cobaugh, C.W., Gutell, R.R.: Evaluation of the suitability of free-energy minimization using mearest-neighbor energy parameters for rna secondary structure prediction. BMC Bioinformatics **5** (2004) 105
8. Mathews, D., Turner, D.: Dynalign: An algorithm for finding the secondary structure common to two RNA sequences. J. Mol. Biol. **317** (2002) 191–203
9. Sankoff, D.: Simultaneous solution of RNA folding, alignment and protosequence problems. SIAM J. Appl. Math. **45** (1985) 810–825
10. Masoumi, B.: A dynamic programming algorithm for the simultaneous alignment and structure prediction of three RNA sequences. Master's thesis, School of Information Technology and Engineering, Faculty of Engineering, University of Ottawa (2005)
11. Sprinzl, M., Horn, C., Brown, M., Ioudovitch, A., Steinberg, S.: Compilation of tRNA sequences and sequences of tRNA genes. Nucl. Acids Res. **26** (1998) 148–153
12. Sprinzl, M., Vassilenko, K.S.: Compilation of tRNA sequences and sequences of tRNA genes. http://www.uni-bayreuth.de/departments/biochemie/trna (2003)
13. Gutell, R.R.: Comparative RNA web site. http://www.rna.icmb.utexas.edu (2004)
14. Cannone, J.J., Subramanian, S., Schnare, M.N., Collett, J.R., D'Souza, L.M., Du, Y., Feng, B., Lin, N., Madabusi, L.V., Muller, K.M., Pande, N., Shang, Z., Yu, N., Gutell, R.R.: The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. BMC Bioinformatics **3** (2002)
15. Cannone, J.J., Subramanian, S., Schnare, M.N., Collett, J.R., D'Souza, L.M., Du, Y., Feng, B., Lin, N., Madabusi, L.V., Muller, K.M., Pande, N., Shang, Z., Yu, N., Gutell, R.R.: The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs: Corrections. BMC Bioinformatics **3** (2002)
16. Gorodkin, J., Stricklin, S.L., Stormo, G.D.: Discovering common stem-loop motifs in unaligned RNA sequences. Nucl. Acids Res. **29** (2001) 2135–2144

Clustering Using Adaptive Self-organizing Maps (ASOM) and Applications

Yong Wang[1], Chengyong Yang[1], Kalai Mathee[2], and Giri Narasimhan[1]

[1] Bioinformatics Research Group (BioRG),
School of Computer Science, Florida International University,
Miami FL 33199, USA
{cyang01, giri}@cs.fiu.edu
[2] Department of Biological Sciences, Florida International University,
Miami FL 33199, USA
matheek@fiu.edu

Abstract. This paper presents an innovative, adaptive variant of Kohonen's self-organizing maps called ASOM, which is an unsupervised clustering method that adaptively decides on the best architecture for the self-organizing map. Like the traditional SOMs, this clustering technique also provides useful information about the relationship between the resulting clusters. Applications of the resulting software to clustering biological data are discussed in detail.

1 Introduction

In today's data-driven world, it has become increasingly important to analyze large amounts of data in order to extract information from it. Such data analysis is now an integral part of genomic and proteomic studies. Data analysis methods can be either exploratory or confirmatory, based on the availability of appropriate models for the data source. Cluster analysis, a dominant technique of exploratory data analysis [1], aims to group a collection of objects into subsets or "clusters", such that those within each cluster are more closely related to one another than objects assigned to different clusters [2].

Data clustering schemes can be described in terms of the following three steps: (1) feature selection or extraction, (2) similarity (or dissimilarity) computation, and (3) clustering. Clustering methods can be broadly classified as hierarchical or partitioning. Strategies for hierarchical clustering divide into two basic paradigms: agglomerative (bottom-up) and divisive (top-down) [2]. Typical partition clustering algorithms include: K-means clustering, and Self-Organizing Maps (SOM). K-means is an iterative descent clustering algorithm [3], while the SOM can be viewed as a constrained version of K-means clustering or a neural network [4, 5]. SOMs have the advantage that it is possible to easily display the output as a two dimensional grid of samples.

The main drawback of the existing algorithms is that they require either specifying the number of clusters in advance (K-means clustering and SOM), or leave the

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 944 – 951, 2005.
© Springer-Verlag Berlin Heidelberg 2005

decision of the number of clusters to the user (hierarchical clustering). To address this shortcomings of fixing *a priori* the number of clusters, Pelleg and Moore proposed a new algorithm called X-means [6], a modification of the traditional K-means algorithm. Instead of specifying the number of clusters in advance, X-means only requires a range of the number of clusters, and then searches the space of cluster locations and the number of clusters using the Bayesian Information Criterion (BIC) measure.

In this paper, we propose a new clustering algorithm: Adaptive Self-Organizing Maps (ASOM), which corresponds to SOMs in the same way that X-means does to K-means. The architecture of the SOM is adaptively modified using a variant of the BIC measure. ASOM only requires a range of architectures to be specified in advance. Our results showed that ASOM is an efficient and reliable tool for cluster analysis of biological data. It turns out to be much more stable than X-means, and as with SOMs, also reveals potential neighborhood relationships among the resulting clusters.

It is well known that it is difficult to ascertain the validity of inferences drawn from the output of these clustering applications. We discuss simple tools that efficiently display information about significant clusters.

1.1 Concepts and Definitions

Samples (or patterns) for clustering are represented as a vector of d measurements or features, i.e., as a point in *d*-dimensional space: $x = (x_1, \ldots x_d)$. A cluster is simply a set of samples. The centroid of a cluster is denoted by μ, and is obtained by doing the coordinate-wise averaging of the points in the cluster. Let N denote the number of clusters. A distance measure or a similarity measure is a metric on the feature space used to quantify the similarity of the samples. Any set of n samples can be viewed as an $n \times d$ matrix. Thus the input to a cluster analysis is an ordered pair (D, s), where D is the matrix of samples and s is the similarity measure. The output from the clustering algorithm is a partition $! = \{G_1, \ldots, G_N\}$, where the clusters G_k, $k = 1, \ldots, N$, are subsets of D, such that $G_1 \bigcup G_2 \cdots \bigcup G_k = D$, and $G_i \bigcap G_j = \Phi, i \neq j$.

1.2 Similarity Measures

A measure of similarity between samples is fundamental to the definition of a cluster, and the quality of clustering depends on its choice. The most popular metric for continuous feature spaces is the *Minkowski metrics*, defined as follows:

$$d_p(X_i, X_j) = \left(\sum_{k=1}^{d} \left| x_{i,k} - x_{j,k} \right|^p \right)^{1/p} = \left\| X_i - X_j \right\|_p$$

When $p = 2$, this is the Euclidean distance metric. Other popular similarity measures include: *Pearson Correlation Coefficient*, *Rho*, *Dice*, *Jaccard*, *Simpson*, and others.

1.3 Evaluating Clusters

Clustering is considered difficult because data can reveal clusters with different shapes and sizes in a d-dimensional feature space. To compound the problem further, the number of clusters in the data often depends on the resolution (fine vs. coarse) with which we view the data. The examples in Figures 1(a) and (b) show that it is possible to get two different sets of clusters depending on whether one wants 3 clusters or 4. Both sets of clusters appear to be equally reasonable, making it necessary to evaluate clustering results objectively and quantitatively [7]. If, for the given data set, *a priori* grouping information is available, then *entropy* can be used to evaluate the clustering results. However, its chief limitation is that it can only be used to compare clusters with the same architecture (i.e., same number of clusters).

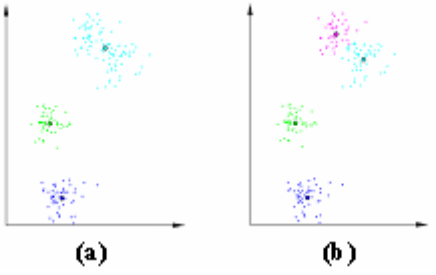


Fig. 1. Input can be partitioned into 3 or 4 clusters

In practice, we may not have *a priori* grouping information. In such cases, the *Bayesian Information Criterion* (BIC) measure can be used to compare the clustering results [8]. The BIC measure is based on the maximization of a log-likelihood score [2]. For X-means and the proposed algorithm, ASOM, the BIC measure is also used as the criterion for model selection. For a given model M, the BIC measure is given by $BIC(M) = l(D) - P / 2 \cdot (\log R)$, where l is the log-likelihood of the data according to model M taken at the maximum likelihood point, and P is the effective number of parameters. Further details may be found in Pelleg and Moore [6]. For finite samples, BIC often chooses simple models to avoid placing heavy penalty on complexity. As the size of samples increases, the probability that the BIC measure favors the correct model also increases [2].

2 Adaptive Self-organizing Maps (ASOM)

One of the shortcomings of SOM algorithm is its fixed network architecture [9]. For a large dataset, it is very difficult to guess the right architecture. This has motivated the development of a number of adaptive variants.

Growing Grid introduces the notion of a *resource* [10]. This is associated with each unit and used to gather statistical information, which is then used in each adaptation step, to decide where in the map a new row or column is to be inserted. Performance of the Growing Grid is superior to conventional SOM [11]. Since its parameters are constant, the user does not need to choose a “cooling schedule”, as in the conventional SOM. However, when the map grows larger, the algorithm is likely to split more than necessary.

The *Growing Hierarchical Self-Organizing Map* (GH-SOM) [9] uses a hierarchical structure with multiple layers, where each layer consists of a number of independent SOMs. Every cell in an SOM from one layer may be added to the next layer of the

hierarchy. The adaptation steps are similar to that of Growing Grid, with the difference that it uses a decreasing learning rate and a decreasing neighborhood radius. The mean quantization error (MQE) of the map is used to decide if a new level of the hierarchy is to be created. Due to its hierarchical structure, GH-SOM shows a lot of structural detail. But the parameters remain hard to choose.

2.1 Algorithm Adaptive Self-organizing Map (ASOM)

Given lower and upper bounds on the number of columns and rows (col_{min}, row_{min}, col_{max}, and row_{max}, respectively), ASOM starts with the smallest architecture. The steps in each iteration are described below.

1. Run SOM algorithm on current architecture: $colcurr \times rowcurr$.
2. Split each "parent" column (and row), and run SOM of architecture $2 \times rowcurr$ on only the data belonging to the parent. Select the architecture that improves the BIC score by the largest amount.
3. If maximum of rows or columns are exceeded, then stop and report the best scoring model during the search.
4. Else update *colcurr* and *rowcurr*. Go to step 1.

In the splitting step, the algorithm decides how to update the architecture and the cluster centers. The algorithm considers every column separately and checks the value of splitting it into two columns. The rows are also considered in a similar fashion. The architecture with the highest BIC score is then chosen for the next iteration. The algorithm continues until the upper bound on the number of rows or columns is reached. The map with the highest BIC score is finally reported.

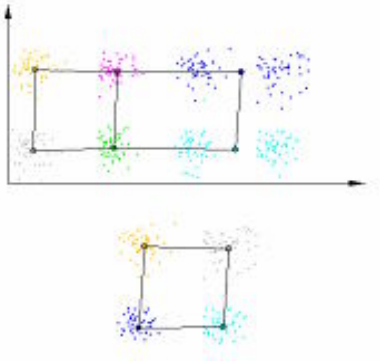


Fig. 2. An example of 2×3 grid being updated to a 2×4 grid. The upper part shows a set of points partitioned into 6 clusters organized into a 2×3 grid. The last column is further partitioned as shown in the lower figure to obtained a 2×4 grid

Figure 3 shows the location of the cluster centers as the ASOM algorithm progresses for a specific chosen architecture. The initial cluster centers are selected at random, and the structure of the network becomes visible after several iterations. As the algorithm progresses, the cluster centers spread out to make meaningful clusters. Figure 4 shows the optimal clustering obtained for each chosen grid architecture. In this example, the best architecture was a 5×5 grid, which had the largest BIC value among the architectures investigated for that data set.

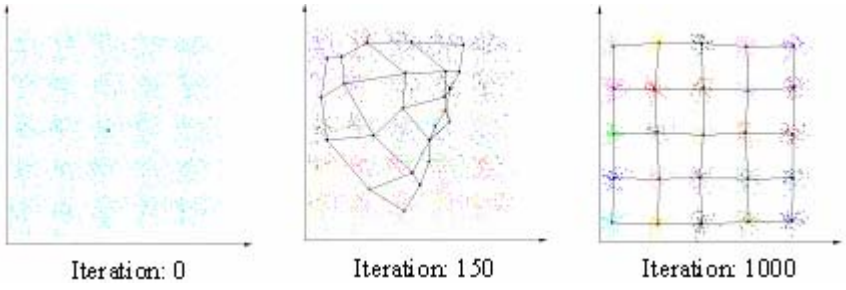


Fig. 3. The positions of the cluster centers (or prototypes) during the execution of the ASOM algorithm for a specific architecture (5×5 grid architecture). More details in [15]

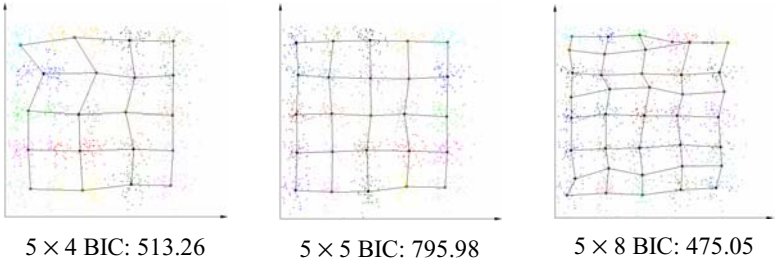


Fig. 4. Experiments with data consisting of points in two-dimensional space are shown above. The maximum number of rows and columns were set to 8. The reference vectors are shown as large black dots, and the neighboring cluster centers are connected by an edge. The numbers at lower right-hand corner indicate the architecture and corresponding BIC score. More details in [15]

3 Experiments and Results

Four clustering algorithms, *K*-means, *X*-means, SOM, and ASOM, were implemented in Java. In order to visualize the whole process, a visualization package was also implemented using Java Swing.

3.1 Rat Central Nervous System (CNS) Dataset

The rat CNS data set, from Wen *et al.*, contains the expression levels of 112 genes during rat central nervous system development over nine time points [12]. As suggested by Wen *et al.*, the raw data set was normalized by the maximum expression level of each gene, and then augmented with slopes (differences between consecutive time points) to take into account offset but parallel patterns. The dataset is a matrix with 112 genes and 17 conditions after the above preprocessing.

The dataset was provided as input to the ASOM software. The best BIC measure for this set of data was obtained with the 2×3 grid architecture. Each grid has one

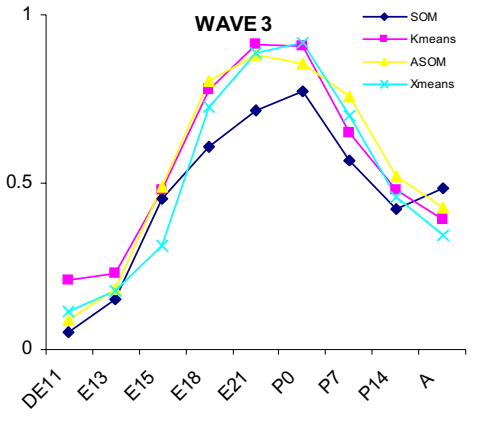


Fig. 5. Mean normalized gene expression levels of gene clusters generated by ASOM, SOM, K-means, X-Means

non-empty cluster, and thus the data set was clustered into six clusters, Wave 1, Wave 2, Wave 3, Wave 4, Constant, and Other [12]. Three other clustering methods in our software were also tested with the following parameter settings: 6 for the number of clusters in K-means and 2×3 for grid architecture in SOM. In order to further compare clustering results, mean normalized expression levels of all gene clusters were plotted. The mean expression plot for one gene cluster (wave 3) is shown in Figure 5 (Plots of all six clusters provided in supplemental website [15]). Average profiles from the four approaches look very similar across all the six clusters. The clusters capture majority significant genes when compared with results in Wen *et al.*

3.2 Yeast Cell Cycle Dataset

The ASOM algorithm was also tested on the yeast cell cycle data of Spellman *et al* [13]. A total of 799 genes were identified as being regulated by cell cycle, and were used as input. To assess the classification capability of the ASOM clusters, gene ontology information was used to evaluate whether the clusters have significant enrichment of one or more function groups (below ontology level 2); this was done using GoMiner [14]. Table 1 shows details of 5 typical clusters with enriched functional groups. For example, cluster 1, with 42 genes, was enriched by DNA and nucleic-acid binding genes. Enriched function groups included helicase activity, cytokinesis, transferase activity, transcription regulator activity and others, suggesting that the ASOM clusters are biologically meaningful (Details of all 8 clusters provided in supplemental website [15]).

3.3 Experiments with Synthetic Data

Random data sets were generated with coordinates from the Gaussian distribution. Two types of data sets were generated. The first one consisted of random point sets where the cluster centers were organized into a pre-specified number of rows and columns. An example of such a data set was shown in Figure 4. The second one consisted of randomly generated point sets where the cluster centers were manually provided by a user. The results showed that the ASOM algorithm was extremely successful in identifying the clusters that were present in the clustered synthetic data that was generated.

Table 1. Enrichment of ASOM clusters by GO function category

Cluster	# of Genes	Enriched functional category (total genes)	Clustered genes	-log10 (p-value)
1	42	nucleic acid binding (37) DNA binding (27)	16 15	12 15
2	130	cell proliferation (77) cell cycle (66)	34 33	9 10
3	20	cell proliferation (77) cytokinesis (14)	10 6	5 6
4	53	helicase activity (17) DNA helicase activity (16)	11 8	9 5
5	26	amino acid metabolism (26)	10	9

4 Discussion

We have developed a new, adaptive variant of Kohonen's self-organizing maps. We apply it to two different gene expression datasets. The proposed adaptive SOM permits the detection of the best architecture for the self-organizing map effectively. The software package provides effective visualization tools to facilitate the analysis. The software is available from the authors upon request.

Supplemental Website: http://biorg.cs.fiu.edu/ASOM

Acknowledgements

Research of GN was supported by NIH Grant P01 DA15027-01.

References

1. Jain, A.K., M.N. Murty, and P.J. Flynn, Data Clustering : A Review. ACM Compting Surveys, 1999. **31**(3): p. 264-323.
2. Hastie, T., R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer Series in Statistics. 2001, New York: Springer. 583.
3. Nagy, G., State of the art in pattern recognition. Proc. IEEE, 1968. **56**: p. 836-862.
4. Kohonen, T., Self-organized formation of topologically correct feature maps. Biological Cybernetics, 1982: p. 43.
5. Kohonen, T., Self-Organizing Maps. 1995.
6. Pelleg, D. and A. Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. in Proceedings of the Seventeenth International Conference on Machine Learning. 2000.

7. Jain, A.K. and R.C. Dubes, Algorithms for Clustering Data. Prentice-Hall advanced reference series. 1988: Prentice-Hall, Inc., Upper Saddle River, NJ.
8. Schwarz, G., Estimating the dimension of a model. Ann. Statist., 1978. **6**(2): p. 461-464.
9. Dittenbach, M., D. Merkl, and A. Rauber. The Growing Hierarchical Self-Organizing Map. in Proc. Intl. Joint Conf. on Neural Networks (IJCNN'00). 2000.
10. Fritzke, B., A growing neural gas network learns topologies, in Advances in Neural Information Processing Systems 7, T.K. Lean, Editor. 1995, MIT Press: Cambridge MA. p. 625-632.
11. Fritzke, B., Kohonen feature maps and growing cell structures - a performance comparison, in Advances in Neural Information Processing Systems 5, J. Cowan, Editor. 1993, Morgan Kaufman Publishers: San Mateo, CA. p. 123-130.
12. Wen, X., et al., Large-scale temporal gene expression mapping of central nervous system development. Proc Natl Acad Sci U S A, 1998. **95**(1): p. 334-9.
13. Spellman, P.T., et al., Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Mol Biol Cell, 1998. **9**(12): p. 3273-97.
14. Zeeberg, B.R., et al., GoMiner: a resource for biological interpretation of genomic and proteomic data. Genome Biol, 2003. **4**(4): p. R28.
15. Additional results and supplemental information for ASOM. http://biorg.cs.fiu.edu/ASOM

Experimental Analysis of a New Algorithm for Partial Haplotype Completion

Paola Bonizzoni[1,3], Gianluca Della Vedova[2,3],
Riccardo Dondi[1,3], and Lorenzo Mariani[1,3]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione
[2] Dipartimento di Statistica
[3] Univ. Milano–Bicocca, Via Bicocca degli Arcimboldi 8, 20126 Milano(Italy)
bonizzoni@disco.unimib.it, riccardo.dondi@unimib.it

Abstract. This paper deals with the computational problem of inferring complete information on haplotypes from haplotypes with missing data. This problem is one of the main issues in *haplotyping*, as the current DNA sequencing technology often produces haplotypes with missing bases and thus the complete information on haplotypes has to be inferred through computational methods. In this paper we propose a new algorithmic approach to the problem that assumes both the Coalescent and the Minimum Entropy models and we provide an experimental analysis relating it to the previously investigated approaches. In particular, the reconstruction of a perfect phylogeny from haplotypes with missing data is addressed.

1 Introduction

After the completion of the Human Genome Project, one of the main goals of Molecular Biology is the understanding of the relationship between variants in the human genome sequence and many common diseases influenced by multiple genetic and environmental factors. The main genetic differences among individuals concern about ten million single DNA sequence sites, referred to as Single Nucleotide Polymorphisms (*SNP*s), in which usually two bases are observed across the population. Therefore, when studying differences among human DNA sequences, only the bases at SNP sites are considered and other bases can be ignored. The sequence of the bases that an individual presents in all or some of his SNP sites is called a *haplotype*.

Recent studies [4] have shown that the human DNA sequence can be subdivided into blocks that have been transmitted through evolution without recombination, and that within each block a few (2-5) distinct common haplotypes that cover about 70-90% of the total observed haplotypes. The National Institutes of Health has launched the Haplotype Map Project [9], whose goal is the identification of the haplotype blocks and the common haplotypes in the genome of different human populations. This project has encouraged the development of computational methods to address several issues and indeed *haplotyping* [1] is a research topic which poses new challenging questions to computer scientists.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 952–959, 2005.
© Springer-Verlag Berlin Heidelberg 2005

A main issue in haplotyping is the inference of complete haplotypes from haplotypes with missing data as the current DNA sequencing technology often produces haplotypes with missing bases at some positions. Combinatorial methods based on inference rules represent a common approach to face this problem [7]. These methods usually use inference rules based on assumptions on evolutionary model for human haplotypes or statistical analysis of haplotypes frequency. Two of the main patterns adopted for this problem are the *coalescent model* (shortly CM) and the *minimum entropy model* (shortly MEM). The coalescent model, proposed in [5] describes the evolutionary relationships between haplotypes, assuming that no recombination occurs between haplotypes and that at each SNP site a mutation can happen only once during evolution. The minimum entropy model, proposed in [6], assumes that the frequency distribution of the haplotypes within the population has a small entropy.

Some existing algorithms assume either CM or MEM. Algorithms assuming both of them, require some strict hypotheses to be satisfied (for example, knowing the phylogenetic tree of the haplotypes) [7]. In this paper we propose a new algorithm that assumes both CM and MEM, requiring only one complete haplotype to be known.

Our algorithm consists of two phases. First, it produces a completion of the partial haplotypes that fits CM, implementing a variant of the "Naïve Algorithm A" of [10]. Then, it executes a local optimization heuristic. In particular, two different procedures have been designed, implemented and tested for this second phase: a greedy improvement procedure and a Kernighan-Lin heuristic style.

Various experiments have been performed in order to study the performance of the algorithm. Experimental results show that the amount of data incorrectly completed is a very low percentage with respect to the amount of total missing data, suggesting that both the greedy improvement heuristic and the Kernighan-Lin heuristic are good approaches for the local optimization of the solution, especially when they are combined. The performance of an algorithm that adopts either the two heuristics has been compared with those of other effective softwares [7, 6, 3]. The results of the comparison are encouraging, showing the effectiveness of the new approach proposed.

2 The Partial Haplotype Completion Problem

In diploid organisms like humans only two different bases are observed in each SNP site, leading to a natural $\{0, 1\}$-representation. Actually, current sequencing technology often produces DNA sequences with some missing bases at some position. Therefore, we must adopt a $\{0, 1, ?\}$-representation to denote the values assumed by each SNP site, where '?' represents the lack of data.

Let $S = \{s_1, s_2, ..., s_n\}$ denote a set of n individuals and let $C = \{c_1, c_2, ..., c_m\}$ denote a set of m SNP sites observed in a specified block of the human genome. Then, *a complete haplotype* over the SNP sites in C is a $\{0, 1\}^m$-vector. A *partial haplotype* over the SNP sites in C is a $\{0, 1, ?\}^m$-vector. A partial haplotype ph is said to be *compatible* with a complete haplotype ch if and only if $ph[i] \neq$ '?'

implies that $ch[i] = ph[i]$, for $i = 1, 2, ..., m$. Given a matrix K, we denote with $K(i)$ the i-th row of K and with K_{ij} the entry corresponding to the i-th row and the j-th column of K. The set of partial haplotypes associated to the set S of individuals is represented by a $\{0, 1, ?\}$-matrix $\mathcal{A}$, called *partial haplotypes matrix*, where each row $\mathcal{A}(i)$ represents a partial haplotype for individual $s_i \in S$. A $\{0, 1\}$-matrix $\mathcal{M}$ is a *complete haplotypes matrix* if it is the completion of $\mathcal{A}$ such that each row $\mathcal{A}(i)$ is compatible with $\mathcal{M}(i)$, for $i = 1, \ldots, n$.

In the partial haplotype inference we assume two models commonly adopted when dealing with this problem, the *coalescent model* and the *minimum entropy model*, described below.

The coalescent model. Recent studies [4] have shown that the human genome can be partitioned into blocks that have been transmitted without showing evidence for recombination. Moreover, recurrent mutations at the same SNP site are rare events in the evolutionary history. Thus, we can assume that the common haplotypes observed in the human population within a genome block fits the CM[5]. The CM describes the evolutionary relations between haplotypes assuming that: (i) there is no recombination between haplotypes and (ii) at each SNP site a mutation can happen only once. By the assumptions of the model, a haplotype with a mutation at a SNP site is a descendant of the ancestor haplotype h_i in which the mutation first occurred, while a haplotype without the mutation cannot be a descendant of h_i. This implies that the evolutionary history of the collected haplotypes is described by a phylogenetic tree known as *perfect phylogeny*.

The problem of finding a completion of a partial haplotypes matrix so that the haplotypes in the complete matrix fit the CM is NP-hard [11]. Nevertheless, it becomes tractable when at least one complete haplotype is known [10]: in this case it is possible to change the {0,1}-representation of the bases at each SNP site in such a way that the known complete haplotype is represented by the all-zero vector. Indeed, in terms of evolutionary tree, this corresponds to fix the known all-zero haplotype as the root of the perfect phylogeny, which becomes the so-called *directed perfect phylogeny*. In what follows we assume that the all-zero vector always belong to the perfect phylogeny, i.e. we adopt the directed or rooted model.

Let $\mathcal{A}$ be a partial haplotypes matrix then, given two columns j_1, j_2 with $1 \leq j_1 < j_2 \leq m$, the *set of the valid pairs* for j_1 and j_2 is the set $VP(j_1, j_2) \subseteq \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ such that $(x, y) \in VP(j_1, j_2)$ if and only if there exists the pair a_{ij_1}, a_{ij_2} of values in a row i, with $1 \leq i \leq n$, such that $(a_{ij_1}, a_{ij_2}) = (x, y)$.

A characterization of the matrices fitting the CM has been given in [7]. Let $\mathcal{M}$ be a complete haplotype matrix, then $\mathcal{M}$ fits the CM if and only if for every pair of columns $j_1, j_2 \in \{1, 2, ..., m\}$ the set of the *valid pairs* $VP(j_1, j_2)$ is such that $\{(0, 1), (1, 0), (1, 1)\} \nsubseteq VP(j_1, j_2)$.

The minimum entropy model. The minimum entropy model is mainly based on a parsimony principle suggested by the fact that in many real-life situations a frequency distribution tends to have a very low entropy value [7, 6].

Assume that a collection of n partial haplotypes $PH = \{ph_1, ph_2, \ldots, ph_n\}$ of length m is given: these are represented by a partial halpotypes matrix $\mathcal{A}$. Let $\alpha: PH \rightarrow \{0,1\}^m$ be the *completion function* that assigns each partial haplotype $ph \in PH$ to a complete compatible haplotype $\alpha(ph) \in \{0,1\}^m$. Thus, let $H = \{\alpha(ph) : ph \in PH\} = \{h_1, h_2, ..., h_k\} \subseteq \{0,1\}^m$ be the set of the distinct complete haplotypes assigned to the partial haplotypes in PH. Let f_i be the number of partial haplotypes in PH that are assigned to the complete haplotype $h_i \in H$: $f_i = |\{ph \in PH : \alpha(ph) = h_i\}|$ for every $i = 1, 2, ..., k$, with $f_1 + f_2 + \ldots + f_k = n$. Thus $(f_1/n, f_2/n, ..., f_k/n) = (p_1, p_2, ..., p_k)$ is the distribution of the relative frequencies of the complete haplotypes assigned to the partial haplotypes in PH.

Observe that the haplotypes $\alpha(ph_1), \alpha(ph_2), ..., \alpha(ph_n)$ give the rows of a complete haplotypes matrix $\mathcal{M}$ of size $n \times m$ that is the completion of the matrix $\mathcal{A}$. Then the entropy of $\mathcal{M}$ is the entropy of the relative frequency distribution $(p_1, p_2, ..., p_k)$, defined as $ENT(\mathcal{M}) = ENT(p_1, p_2, ..., p_k) = \sum_{i=1}^{k} \left(-p_i \log_2 p_i\right)$.

The MEM assumes that the most likely valid completion of the partial haplotypes is the one that minimizes the entropy value $ENT(\mathcal{M})$; it is proved that finding such a completion is an APX-hard problem [6].

3 An Algorithm for Partial Haplotype Completion

We propose a new algorithm for the completion of a collection of partial haplotypes, represented as a $\{0, 1, ?\}$-matrix, assuming that both the CM and the MEM hold. More precisely, the use of the two models lead to search for the partial haplotypes completion that determines the minimum entropy frequency distribution, within the set of the completions that fit the CM. Formally, each instance of the *Partial Haplotype Completion Problem* is a partial haplotype matrix $\mathcal{A}$ of size $n \times m$, and the goal is to find (if it exists) a matrix $\mathcal{M}$ of size $n \times m$ that is a completion of $\mathcal{A}$ such that $\mathcal{M}$ fits the CM and $ENT(\mathcal{M})$ is minimum.

We do not know whether there is a polynomial-time algorithm to find such a completion matrix $\mathcal{M}$, similarly to the case in which the evolutionary history of the complete haplotypes is known [7]. Thus, we propose a heuristic algorithm for the problem composed of two phases: first we compute an initial completion that fits the CM, if it is possible, then such completion is modified in order to find a completion that still fits the CM and that determines a smaller entropy.

For the first phase, we design a variant of the "Naïve Algorithm A" proposed in [10] for the incomplete directed perfect phylogeny reconstruction and that we denoted by (*pp*). When this algorithm is invoked on a $\{0, 1, ?\}$-matrix of size $n \times m$, its computational complexity is $O(hnm)$, where $h \leq \min\{m, n\}$.

The results produced in the first phase are refined in the second phase of the algorithm. In particular we use two local optimization heuristic procedures: a greedy heuristic (gh) and a Kernighan-Lin style heuristic (klh), described below. In Section 4 we compare the performance of the two procedures when executed separately or combined.

The greedy heuristic. The greedy heuristic procedure (gh) is inspired by the greedy algorithm proposed in [6], whose goal is finding a minimum entropy set of complete haplotypes compatible with a given collection of partial haplotypes. Given a partial haplotype matrix $\mathcal{A}$ and a completion matrix $\mathcal{M}$ of $\mathcal{A}$ produced by (pp), the procedure identifies the distinct complete haplotypes of $\mathcal{M}$ and calculates their frequencies. Then, it simply takes each distinct complete haplotype h of $\mathcal{M}$ starting from the most frequent one and ending with the least frequent one, and assigns it to the not-yet-assigned compatible partial haplotypes of $\mathcal{A}$, defining a new completion matrix $\mathcal{M}'$. The goal is to concentrate the maximum possible number of haplotypes into large sets, reducing the entropy of their frequency distribution.

The procedure runs in $O(n^2m)$ time, since it takes $O(n^2m)$ time to detect the distinct complete haplotypes of $\mathcal{M}$ and their frequencies, $O(n)$ time to rearrange them in growing order of frequency (with counting sort algorithm) and $O(n^2m)$ time to assign these complete haplotypes to partial haplotypes of $\mathcal{A}$. In some cases, a great set of identical haplotypes can be obtained only considering complete haplotypes different from the ones obtained by applying (pp) and for this goal we have designed a Kernighan-Lin heuristic (klh)[8].

The Kernighan-Lin heuristic. The Kernighan-Lin style heuristic we propose applies a local improvement of the solution based on the iterative inversion (from '0' to '1', or from '1' to '0') of the value of an entry m_{ij} of the completion matrix $\mathcal{M}$ that corresponds to a missing data in $\mathcal{A}$, that is $a_{ij} =$ '?'. Note that this inversion can be applied only if the resulting matrix $\mathcal{M}'$ still fits the CM. Thus, before operating the inversion we must check, with respect to the Valid Pairs Theorem, the valid pairs of each pair of columns of the matrix $\mathcal{M}'$ that we would obtain performing the inversion. This operation takes $\Theta(nm^2)$.

Moreover, we must design an efficient function that, given a completion matrix $\mathcal{M}$ of matrix $\mathcal{A}$, the complete haplotypes of $\mathcal{M}$ and their frequency distribution, calculates the entropy of the frequency distribution that we would obtain inverting a specific entry m_{ij} (with $a_{ij} =$ '?'). Identifying the complete haplotypes of $\mathcal{M}$ and determining their frequency distribution takes $O(n^2m)$ time. Calculating the new entropy of the frequency distribution, instead, takes $O(nm)$ time, since comparing two haplotypes of length m takes $O(m)$ time and such a comparison is performed for all the complete haplotypes, that are at most n.

The total number of iterations cannot be determined a priori, since it depends on the random selections at execution time, however the core of the algorithm runs in $O(n^2m^2)$ time.

4 Results

Our algorithm has been tested on input data coming from the common haplotypes found in the blocks 1–7 and 10–11 of a region of chromosome 5p31, as defined by Daly et al. in [2]. Starting with this data, we have constructed 9 complete haplotypes $\{0,1\}$-matrices (matrices $\mathcal{C}$), composed of 86–97 rows and 5–31 columns and then, adding missing data to them, almost 40 partial haplotypes

Algorithm 1: Kernighan Lin Heuristic (klh)

Data: a $\{0,1,?\}$-matrix $\mathcal{A}$ of size $n \times m$, an integer parameter *max_iter*; a real number parameter *base*
compute the distinct complete haplotypes of $\mathcal{M}$, and the frequency distribution;
$\mathcal{B} = \mathcal{M}$; // $\mathcal{B}$=*so-far best found completion*;
min_entropy = entropy of $\mathcal{M}$;
while *(number of iterations performed without improvement* $<$ *max_iter)* **do**
  foreach *'?'-entry* a_{ij} *of* $\mathcal{A}$ **do**
    if *the inversion of* m_{ij} *in* $\mathcal{M}$ *is admissible* **then**
      new_entropy = entropy of $\mathcal{M}$ with m_{ij} inverted;
      $p(a_{ij}) = base^{-new_entropy}$;
      if *new_entropy* $<$ *min_entropy* **then**
        $\mathcal{B} = \mathcal{M}$; // *an improvement has been obtained*;
        invert b_{ij};
        min_entropy = *new_entropy*;
    else $p(a_{ij}) = 0$;
  end
  with probability $p(a_{ij})$ for each '?'-entry a_{ij} of $\mathcal{A}$, select a '?'-entry $a_{\hat{i}\hat{j}}$;
  invert $m_{\hat{i}\hat{j}}$ in $\mathcal{M}$;
  update the number of occurrences of all valid pairs for each column pair of $\mathcal{M}$;
  update the distinct complete haplotypes of $\mathcal{M}$, and the frequency distribution;
end
Result: $\mathcal{B}$

$\{0,1,?\}$-matrices (matrices $\mathcal{A}$) whose rate of unknown data varied from 1% to 50% have been obtained.

The completion of each of these $\{0,1,?\}$-matrices (matrices $\mathcal{M}$) has been computed by adopting four different approaches: executing the single procedure (pp); executing (pp) and (gh); executing (pp) and (klh); executing (pp), followed by the execution of (gh) and then (klh) (we denote this algorithm as (pp)+(gh)+(klh)). For the executions of (klh), different values have been assigned to its two parameters: the base of the exponential used to calculate the inversion probability (from 10 to 100) and the maximum number of iterations without improvements (from 5 to 25).

For each completion matrix $\mathcal{M}$ obtained for an incomplete haplotype matrix we have evaluated the error rate, that is the number of entries of $\mathcal{M}$ that are different from the corresponding entries of $\mathcal{C}$, divided by the total number of '?' entries of $\mathcal{A}$, and the entropy difference, that is the difference between the entropy values of matrices $\mathcal{C}$ and $\mathcal{M}$ and are reported in Table 1.

Observe that Table 1 shows that some incomplete matrices admit completion matrices with an entropy lower than that of the complete matrices from which they have been generated; this fact especially happens when the number of sampled individuals is small. The Table also suggests that the greedy heuristic

Table 1. Comparing the entropy values of matrices $\mathcal{C}$ and $\mathcal{M}$

	Entropy differences				Error rates			
Approach	Mean	Min	Max	Std.Dev	Mean	Min	Max	Std.Dev
(pp)	0.078	-0.210	0.414	0.160	4.20%	0.00%	13.63%	0.0420
(pp+klh)	0.043	-0.210	0.414	0.137	3.42%	0.00%	12.81%	0.0400
(pp+gh)	-0.001	-0.210	0.414	0.108	2.64%	0.00%	11.30%	0.0350
(pp+gh+klh)	-0.006	-0.210	0.414	0.106	2.51%	0.00%	11.30%	0.0339

produces better results than (klh), but the best performance is obtained by combining the two algorithms. The recorded execution times have been always under one second (on a 1.60 GHz processor).

In order to evaluate the scalability of the algorithm, we have tested it on 500 artificial $\{0, 1, ?\}$-matrices composed by 50–250 rows and 50–250 columns. Although the average entropy differences and the average error rates show a behaviour that is similar to the one described above, this experiment shows that (klh) strongly affects the execution time of the algorithm (determining a running time up to 600 seconds), when the number of sampled SNP sites is great or when there is an high rate of missing data. Thus, in cases in which the input instance has a very great number of SNP sites or an high rate of unknown data, the use of the (pp+gh) approach seems to be the best solution.

For the above reasons we have tested the (pp+gh) approach on 20 artificial $\{0, 1, ?\}$-matrices of size $10,000 \times 1,000$, varying the rate of unknown data from 1% to 40%. We have obtained an error rate with range from 0% to 0.000005% and an error percentage up to 0.000002%. The error rates are very small, since we have already observed that the error rate tends to quickly decrease as the number of SNP sites grows. Furthermore, the algorithm has required an average execution time of 54 seconds (ranging from 17 to 113 seconds) on a 1.60 GHz processor; this shows that the (pp+gh) approach scales very well to large data sets.

5 Discussion

In this section we remark the differences between (pp+gh+klh) algorithm and other algorithms proposed for the partial haplotype completion problem. First of all we discuss the differences in terms of models and hypothesis assumed. Two algorithms proposed in [7] assume both CM and MEM, but they both require that the perfect phylogeny of the complete haplotypes is known, while our algorithm only requires the knowledge of one complete haplotype.

The algorithm proposed in [6] assumes only the MEM; it is based on a greedy approach, and runs in exponential time when the complete haplotypes are not known and the number of SNP sites is not logarithmic with respect to the number of individuals. Two algorithms proposed in [7], assume only the CM and require the input data to meet the rich data hypothesis, according to which each pair columns contains three valid pairs. Under this hypothesis the problem

of completing the partial haplotypes of $\mathcal{A}$ assuming the coalescent model is polynomial [6]. Next we focus on the comparison between the (pp+gh+klh) algorithm with other published algorithms, based on experimental data.

On real data, when the rate of missing data is no more than 20%, our algorithm has given an average error rate of 0.85% (with range from 0.00% to 6.52%) and an average percentage of errors of 0.17% (with range from 0.00% to 1.30%) with respect to the total number of entries.

We compare these experimental results with those of two algorithms of [7, 3]. Both these algorithms are executed on all the haplotypes found within a block and not only on the common ones, as our algorithm does. Moreover the algorithm proposed in [3] considers as input a collection of genotypes rather than haplotypes, thus it even includes a procedure for the inference of haplotypes from genotypes. Hence, the performance results that we report do not allow a detailed comparison, but they still give a new and interesting point of view.

The greedy algorithm proposed in [7] has been executed on real-data matrices composed by 90–129 rows and 10–31 columns, with missing data rate varying from 0.5% to 10%; it has given average error rates between 2.8% and 8.1%.

The maximum-likelihood-based algorithm proposed in [3] has been executed on real-data matrices composed by 129 rows and 5–31 columns, with a missing data rate of 10.03%; it has given a percentage of errors of 0.23% with respect to the total number of entries, with a running time of a few seconds.

References

1. P. Bonizzoni, G. Della Vedova, R. Dondi, and J. Li. The haplotyping problem: a view of computational models and solutions. *Journal of Computer and Science Technology*, 18:675–688, 2003.
2. M. J. Daly, J. D. Rioux, S. F. Schaffner, and *et al.* High-resolution haplotype structure in the human genome. *Nat. Genet.*, 29(2):229–232, 2001.
3. E. Eskin, E. Halperin, and R. M. Karp. Large scale reconstruction of haplotypes from genotype data. In *Proceedings of the 7th RECOMB*, pages 104–113, 2003.
4. S. B. Gabriel, S. F. Schaffner, and *et al.* The structure of haplotype blocks in the human genome. *Science*, 296:2225–2229, 2002.
5. D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proceedings of the 6th RECOMB*, pages 166–175, 2002.
6. E. Halperin and R. M. Karp. The minimum-entropy set cover problem. In *Proceedings of the 31st ICALP*, pages 733–744, 2004.
7. E. Halperin and R. M. Karp. Perfect phylogeny and haplotype assignment. In *Proceedings of the 8th RECOMB*, 2004.
8. K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.
9. National Institutes of Health. Large-scale genotyping for the haplotype map of the human genome. RFA (Request For Applications) HG-02-005, 2002.
10. I. Pe'er, T. Pupko, R. Shamir, and R. Sharan. Incomplete directed perfect phylogeny. *SIAM Journal on Computing*, 33(3):590–607, 2004.
11. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

Improving the Sensitivity and Specificity of Protein Homology Search by Incorporating Predicted Secondary Structures

Bin Ma, Lieyu Wu, and Kaizhong Zhang

Department of Computer Science, University of Western Ontario,
London, ON N6A 5B7, Canada
{bma, lwu7, kzhang}@csd.uwo.ca

Abstract. In this paper, we improve the homology search performance by the combination of the predicted protein secondary structures and protein sequences. Previous research suggested that the straightforward combination of predicted secondary structures did not improve the homology search performance, mostly because the errors in the structure prediction. We solved this problem by taking into account the confidence scores output by the prediction programs.

1 Introduction

Proteins are essential to all living beings. They are involved in almost every cell function as enzymes and act as structural material of cells. The primary structure of a protein is a sequence of amino acids. The similarity at the sequence level of two proteins is often used to detect protein homology in practice. Tremendous amount of research has been done in the comparison of protein sequences.

For measuring the similarities between pairs of amino acids, amino acid score matrices have been developed, the most popular ones being the BLOSUM matrices [10] and the PAM matrices [9]. Once a score matrix is chosen, the Smith-Waterman algorithm [18] can be used to compute the the optimal sequence alignment of the two protein sequences. The score of the alignment is equal to the sum of the scores between pairs of amino acids at every position of the alignment.

Usually, a protein sequence is compared with every protein sequence in a protein database to find possible homologs. Protein databases are typically large. As a result, it would take too much time to run the Smith-Waterman algorithm to compare the query protein with every protein in the databases. To address this speed issue, researchers have developed heuristic algorithms to quickly identify the possible targets and run Smith-Waterman-like algorithm only on the possible targets. The most famous software program utilizing such a strategy is BLAST [1, 2]. The other popular programs include FASTA [14] and PatternHunter [15, 13].

Besides the speed issue, there are two other important issues, the sensitivity and the specificity, to evaluate the quality of a homology search method.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 960–967, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Sensitivity measures the ability of the programs to find all of the real homologies. The loss of sensitivity comes from two reasons. The first is that programs like BLAST, FASTA and PatternHunter all use heuristic methods that sacrifice the sensitivity to gain speed. The recent PatternHunter II paper uses multiple spaced seeds to achieve close to Smith-Waterman sensitivity with much faster speed [13]. Therefore, this reason of losing sensitivity has been mostly solved.

However, the second reason of losing sensitivity is that the score defined purely by protein sequence alignment may not be able to recognize the homologies at all. Researchers have noticed that when the sequence identity between known protein and their homologs falls below a threshold of around 20%, sequence alignment are not reliable for finding the homologs [3]. This reason of losing sensitivity cannot be solved by any methods that are purely based on sequence comparison.

Specificity measures the ability of the programs of excluding the faked homologies. Because the score defined by purely sequence comparisons is not a very accurate measure of homology, if we set the score threshold low in order to get more real homologs, we end up with picking up many faked homologs with certain sequence similarities.

The secondary structures of proteins have been used to help the homology finding [4, 21]. The theory is that protein structures are well known to be much more conserved than the protein sequences during the evolution. However, because predicted secondary structures contain errors, the straightforward use of the protein secondary structure did not give any improvement [21] on the performance. Hence, either further criteria were needed to determine the real homologs after the initial selection using secondary structures [4], or the alignments need to be done multiple times with different settings of weights on the sequences and structures [21], which reduces the running speed by multiple folds.

Many computational methods such as those in [8, 17, 12] have been developed for predicting the secondary structure from the amino acid sequence of a protein. Some of them, including PSIPRED [12], give a confidence score to each position of the prediction. In this paper we propose to utilize this confidence score in homology searches. We will show that this improves the homology search sensitivity and specificity significantly by extensive test on some data, as well as prove some nice mathematical properties of the proposed scoring methods.

The contributions of this paper include:

(1) New scoring methods that utilize the predicted secondary structures and the prediction confidence;

(2) Software that uses the new scoring method to do homology search.

2 A New Scoring Method for Homology Search

2.1 Protein Secondary Structure

The secondary structure of a protein specifies the local spacial arrangements of the protein sequences. Three general types of local arrangements (secondary

structure units) can be formed. They are the α helix (H), β sheet (E) and coil (C). In the representation of a secondary structure, each amino acid of the protein is labeled with one of the three letters, H, E and C, indicating the amino acid is involved in a unit of the corresponding type. For example, the following is a segment of a protein sequence with its secondary structure:

```
Amino Acid Sequence:        ...VLSATDKSNVKAAWGKVGG...
Secondary Structure:        ...CCCHHHHHHHHHHHHHHCC...
```

The amino acid and its corresponding secondary structure label is called an *aa/ss pair* (amino acid/secondary structure pair) in the paper. Consequently, the sequence and secondary structure together can be regarded as a sequence of aa/ss pairs.

Given an amino acid sequence, a secondary structure prediction program can be used to predict the secondary structure. Many prediction programs including PSIPRED [12] gives a confidence score to the prediction at each position. The confidence score of PSIPRED ranges from 0 to 9. For example, the following is a segment of PSIPRED's output when using a protein sequence as input:

```
...VLSATDKSNVKAAWGKVGG...
...CCCHHHHHHHHHHHHHHCC...
...9889999999999744301...
```

Although PSIPRED outputs a number between 0 to 9 as its confidence score at each position, for convenience, later on we divide this score by 10 to normalize it to be a value between 0 and 1.

2.2 The Proposed Scoring Functions

In [21], the Smith-Waterman algorithm is utilized to align two sequences of aa/ss pairs. And the optimal local alignment score is used as the similarity score between two proteins. More specifically, the author first computed a secondary structure score matrix, M^{ss}, by using a statistical method similar to [10]. The M^{ss} given in [21] is shown in Table 1.

Then an amino acid score matrix, denoted by M^{aa}, is selected to measure the similarity between amino acids. M^{aa} can be set to be one of the BLOSUM matrices or PAM matrices.

Let α and β be the weighted importance of the amino acid and secondary structure in the alignment. Let $\langle a, s\rangle$ and $\langle a', s'\rangle$ be two aa/ss pairs. The similarity score of the two pairs is defined by

$$M(\langle a, s\rangle, \langle a', s'\rangle) = \alpha \times M^{aa}(a, a') + \beta \times M^{ss}(s, s') \quad (1)$$

Table 1. The structure score matrix used by Wallqvist *et. al.* 2000

	H	E	C
H	2	-15	-4
E	-15	4	-4
C	-4	-4	2

Let g_o be the gap open penalty, and g_e be the gap extension penalty. The similarity score of two proteins is defined in [21] by the optimal local alignment score of the two sequences of aa/ss pairs.

In [21], (α, β) is set to be $(1, 0)$, $(0.9, 0.1)$, ..., $(0, 1)$, respectively. The study of [21] demonstrated that if the true secondary structures of the proteins are used, then the performance is maximized for $(\alpha, \beta) = (0.5, 0.5)$, i.e., when putting equal weight at the sequence and secondary structure. However, if the predicted secondary structures are used, then the performance is the best for $(\alpha, \beta) = (1, 0)$, i.e., when not using secondary structures at all.

This is somewhat discouraging for the use of predicted secondary structures. The authors of [21] then proposed to do eleven different alignments for the eleven different values of (α, β) in a row, and use the cumulative homology search results. This indeed showed some degree of improvement when the predicted secondary structures are used. However, this immediately increases the running time by a factor of 10, which is usually not acceptable.

The results in [21] suggest that the true secondary structures can improve the homology search performance. However, the errors in the predicted secondary structures make them not much useful. Therefore, we propose to use the confidence scores output by the secondary structure prediction programs to distinguish the positions with more accurate predictions from the positions with more errors; and put more weight to the more accurately predicted positions in the score computation.

More formally, let $\langle a, s\rangle$ and $\langle a', s'\rangle$ be two aa/ss pairs. Let p and q be the confidence scores of the predictions at the two pairs, respectively. $0 \leq p, q \leq 1$. Then the similarity score of the two pairs is defined by

$$M(\langle a, s\rangle, \langle a', s'\rangle) = M^{aa}(a, a') + p \times q \times M^{ss}(s, s') \tag{2}$$

Once the scores between aa/ss pairs are defined by (2), for any given gap open and gap extension penalty, the standard optimal local alignment score of the two aa/ss sequences is used to be the similarity score between two proteins.

We note that there are ways other than (2) to define the aa/ss score, as follows

$$M(\langle a, s\rangle, \langle a', s'\rangle) = M^{aa}(a, a') + \min\{p, q\} \times M^{ss}(s, s') \tag{3}$$

and

$$M(\langle a, s\rangle, \langle a', s'\rangle) = M^{aa}(a, a') + (\min\{p, q\})^2 \times M^{ss}(s, s') \tag{4}$$

The aa/ss score defined in (2) is very intuitive. However, in the next section we will show that the scores defined by (3) and (4) have important properties that (2) does not have.

2.3 The Strong Similarity Measure

In homology searches, similarity measures are usually used, where identical sequences have the maximized similarity score. The advantage of using similarity measure is that the optimal local alignment can be nontrivially defined. This is

not the case for edit distance because the edit distance [6] of two empty strings give us the optimal value 0.

However, the disadvantage of similarity measure is that there is no Triangular Inequality, which can be very useful for building efficient database indexes such as M-tree [7]. Fortunately, researchers observed that most BLOSUM matrices satisfy the following two properties [20].

$$M(x,y) \leq \min\{M(x,x), M(y,y)\} \tag{5}$$

$$M(x,y) + M(y,z) \leq M(x,z) + M(y,y) \tag{6}$$

The two properties can be regarded as the equivalence of Triangular Inequality in the similarity measure. Once these properties are satisfied, in this paper we call the similarity measure a *strong similarity measure*.

The two properties of a strong similarity measure have been exploited to index protein databases for faster homology searches in previous research such as [20, 11, 19]. It is shown in [20, 11, 19] that if the score matrix is a strong similarity measure, then the optimal local alignment score based on that score matrix is also a strong similarity measure. Therefore, it is desirable to have the score matrices defined in Section 2.2 be strong similarity measures.

Lemma 1. *Let $M(\cdot,\cdot)$ be a strong similarity measure. For any object x, $0 \leq p_x \leq$ be a real number. Then the measure $M'(x,y) = M(x,y) \times \min\{p_x, p_y\}$ is also a strong similarity measure.*

Proof. Proof will be provided in the full version of this extended abstract. □

Lemma 2. *Let $M(\cdot,\cdot)$ be a strong similarity measure. For any object x, $0 \leq p_x \leq$ be a real number. Then the measure $M'(x,y) = M(x,y) \times (\min\{p_x, p_y\})^2$ is also a strong similarity measure.*

Proof. From Lemma 1, we know that $M(x,y) \times (\min\{p_x, p_y\})$ is a strong similarity measure. By using Lemma 1 again, we know that $[M(x,y) \times (\min\{p_x, p_y\})] \times (\min\{p_x, p_y\})$ is a strong similarity measure. □

Lemma 3. *If M_1 and M_2 are strong similarity measures, then $M_1 + M_2$ are strong similarity measure.*

Proof. Trivial. □

Lemmas 1, 2 and 3 show that if the amino acid matrix, M^{aa}, and the secondary structure matrix, M^{ss}, are strong similarity measure, then the scores defined by (3) and (4) are also strong similarity measure. On the contrary, we show in the following that the score defined by (2) may be not.

For conflicting the property of (5), suppose that $M(x,y) = 1$, $M(x,x) = 2$, $p_x = 0.1$, and $p_y = 0.9$. Then $0.09 = M(x,y) \times p_x \times p_y > M(x,x) \times p_x \times p_x = 0.02$

For conflicting the property of (6), suppose that $M(x,y) = -1$, $M(y,z) = -1$, $M(x,z) = -1$, $M(y,y) = 1$, $p_x = 0.9$, $p_y = 0.1$ and $p_z = 0.9$. Then $-0.18 = M(x,y) \times p_x \times p_y + M(y,z) \times p_y \times p_z > M(x,z) \times p_x \times p_z + M(y,y) \times p_y \times p_y = -0.8$.

3 Experiments

We implemented the Smith-Waterman algorithm by using the score schemes defined by (2), (3), and (4), respectively. For comparison purpose, we also implemented the Smith-Waterman algorithm by using the score scheme defined in the following.

$$M(\langle a, s\rangle, \langle a', s'\rangle) = M^{aa}(a, a') + M^{ss}(s, s'). \tag{7}$$

In (7), the predicted secondary structure is used but the confidence scores are not. In all the implementations, the BLOSUM62 matrix is used as the amino acid matrix M^{aa} and the matrix in Table 1 is used as the secondary structure matrix M^{ss}. The gap open penalty $g_o = -12$ and the gap extension penalty $g_e = -2$. We used the PSIPRED [12] program to predict the secondary structures of all proteins that we deal with. The version of PSIPRED we used is 2.3.

The database we used to search in is the release 39 (December 31, 1993) of the PIR database [5], which was downloaded from http://pir.georgetown.edu/. The reason we use this version of the database is to be consistent with [16]. Also, we used the same 54 protein superfamilies as used in [16].

For each superfamily, a representative protein was given in [16]. We use that representative protein as query to search the whole PIR database. Suppose the superfamily has size K, and the top K sequences given by our search has K' sequences in the superfamily. Then $K - K'$ sequences are errors. And the ratio

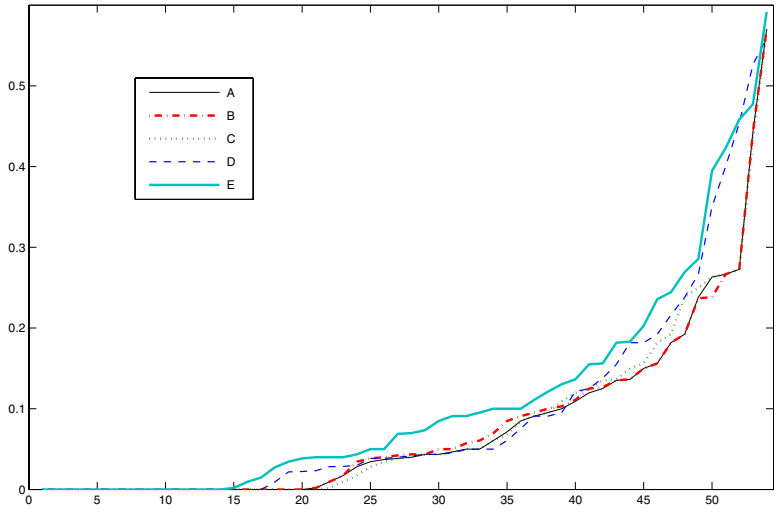


Fig. 1. The EER curve of the five scoring functions on the 54 protein families. A) sequence + weighted structure score as in(2); B) sequence + weighted structure score as in(3); C) sequence + weighted structure score as in(4); D) sequence score only; E) sequence + structure score as in (7)

$\frac{K-K'}{K}$ is called the *equal error rate* (ERR) of the searching method at that superfamily. We then sort the 54 ERR values and plot a ERR curve in Figure 1.

From Figure 1 one can conclude that simply adding the structure score based on the predicted secondary structure will make the performance worse (curve E). This is consistent with the observation in [21]. However, by accounting for the confidences predicted by the secondary structure prediction program, the performance is much better (curves A, B and C).

4 Discussion

We proposed three new formulae, (2), (3), and (4) to combine the amino acid score and secondary structure score in homology search. By using PSIPRED software to predict the secondary structures of both the query protein sequence and the database protein sequences, the combined score gave significantly better sensitivity and specificity than the ordinary searching method which uses only the amino acid sequences. Our research result is surprising because previous research [21] suggested that combining the predicted secondary structures (instead of the true secondary structures) was not better than using sequences only. The difference between our research and the previous ones is that we take into account the confidence scores output by the secondary structure prediction programs.

When the amino acid matrix and the secondary structure matrix are both strong similarity measures, some of the formulae we used ((3), and (4)) give us strong similarity measures.

As future work, the gap open and gap extension penalties need to be tuned for different amino acid matrices and secondary structure matrices. The software programs for the experiments will be made available at http://bif.csd.uwo.ca/.

Acknowledgment

This research was undertaken, in part, thanks to funding from NSERC, PREA, and the Canada Research Chairs Program. We thank Dr. Dong Xu for his recommendation of the PSIPRED secondary structure prediction program.

References

1. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.
2. S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3392, 1997.
3. T. K. Attwood and D. J. Parry-Smith. *Introduction to Bioinformatics*. Prentice Hall, 1999.

4. R. Auroroa and G. Rose. Seeking an ancient enzyme in methanococcus jannaschii using orf, a program based on predicted secondary structure comparisons. *Proc. Natl. Acad. Sci. USA*, 95:2818–2823, 1998.
5. G. D. Barker and W. L. Hunt. Pir-international protein sequence database. *Methods Enzymol*, 266:41–59, 1996.
6. D. Brown, M. Li, and B. Ma. The homology search methods. In L. Wong, editor, *The Practical Bioinformatician*. World Scientific, 2004.
7. P. Ciaacia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Database*, pages 15–26, 1997.
8. J. A. Cuff and G. J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *PROTEINS: Structure, Function and Genetics*, 34:508–519, 1999.
9. M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Prot. Seq. and Struct.*, 5:345–352, 1978.
10. S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.
11. M. Itoh, S. Goto, T. Akutsu, and M. Kanehisa. Fast and accurate database homology search using upper bounds of local alignment. *Bioinformatics*, 2004. To appear.
12. D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292:195–202, 1999.
13. M. Li, B. Ma, D. Kisman, and J. Tromp. PatternHunter II: Highly sensitive and fast homology search. *Journal of Bioinformatics and Computational Biology*, 2(3):417–440, 2004.
14. D. Lipman and W. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227, 1985.
15. B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, March 2002.
16. W. R. Pearson. Comparison of methods for searching protein sequence databases. *Protein Science*, 4:1145–1160, 1995.
17. B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, 232:584–599, 1993.
18. T. Smith and M. Waterman. Identification of common molecular subsequences. *J.Mol.Biol*, 147:195–197, 1981.
19. P. Spiro and N. Macura. A local alignment metric for accelerating biosequence database search. *Journal of Computational Biology*, 11:61–82.
20. A. Stojmirović and V. Pestov. Indexing schemes for similarity search in datasets of short protein fragments. Technical report, 2003. ArXiv e-print DS/0309005.
21. A. Wallqvist, Y. Fukunishi, L. R. Murphy, A. Fadel, and R. M. Levy. Iterative sequence/secondary structure search for protein homologs: comparison with amino acid sequence alignments and application to fold recognition in genome databases. *Bioinformatics*, 16(11):988–1002, 2000.

Profiling and Searching for RNA Pseudoknot Structures in Genomes

Chunmei Liu[1], Yinglei Song[1], Russell L. Malmberg[2], and Liming Cai[1]

[1] Department of Computer Science, University of Georgia, Athens GA 30602, USA
{chunmei, song, cai}@cs.uga.edu
[2] Department of Plant Biology, University of Georgia, Athens GA 30602, USA
russell@plantbio.uga.edu

Abstract. A new method is developed that can profile and efficiently search for pseudoknot structures in noncoding RNA genes. It profiles interleaving stems in pseudoknot structures with independent Covariance Model (CM) components. The statistical alignment score for searching is obtained by combining the alignment scores from all CM components. Our experiments show that the model can achieve excellent accuracy on both random and biological data. The efficiency achieved by the method makes it possible to search for the pseudoknot structures in genomes of a variety of organisms.

1 Introduction

Searching genomes with computational models has become an effective approach to the identification of genes. During recent years, extensive research has been focused on developing computationally efficient and accurate models that can find novel noncoding RNAs and reveal their associated biological functionalities. Functionalities of noncoding RNAs are, to a large extent, determined by the secondary structures they fold into. Secondary structures are formed by bonded base pairs between nucleotides and may remain unchanged over evolution while the nucleotides of a sequence have been significantly modified through mutations. Profiling models based solely on sequence content such as Hidden Markov Model (HMM) [9] may miss important homologies when directly used to search genomes for noncoding RNAs containing complex secondary structures. Therefore, models that can profile noncoding RNAs must include both the content and structural information from the homologous sequences. The Covariance Model (CM) developed by Eddy and Durbin [5] extends the profiling HMM by allowing the coemission of pairing nucleotides on certain states to model base pairs, and introduces bifurcation states to emit parallel stems in the conformation. The CM is capable of modeling secondary structures comprised of nested and parallel stems. However, pseudoknot structures, where at least two structurally interleaving stems are involved, cannot be directly modeled with the CM and remained computationally intractable for searching [1][10][13][14][15].

So far, only a few systems have been developed for profiling and searching for RNA pseudoknots. One example of related work includes ERPIN, developed by Gautheret and Lambert [6][11]. ERPIN searches genomes by sequentially looking for single stem

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 968–975, 2005.
© Springer-Verlag Berlin Heidelberg 2005

loop motifs contained in the noncoding RNA gene. Since ERPIN does not allow the presence of gaps when it performs alignments, it is computationally very efficient. However, alignments with no gaps may miss distant homologies and thus result in a lower sensitivity. Another example is from Brown and Wilson [2]. They proposed a more realistic model comprised of a number of Stochastic Context Free Grammar (SCFG) [5][16] components to profile pseudoknot structures. In their model, different components are used to derive the interleaving stems in a pseudoknot structure. The optimal alignment score of a sequence segment is computed by aligning it to all the components iteratively. The model can be used to search sequences with simple pseudoknot structures efficiently. However, for pseudoknots with more complex structure, more than two SCFG components may be needed and the extension of the iterative alignment algorithm on k components may need to perform $k!$ different alignments in total since all components are treated equally in their model.

In this paper, we propose a new method to search for RNA pseudoknot structures using a model of multiple CMs. Unlike the model of Brown and Wilson, independent CM components are used to profile the interleaving stems in a pseudoknot. Based on the model, we have developed a generic framework for modeling interleaving stems of pseudoknot structures; we propose an algorithm that can efficiently assign stems to components such that interleaving stems are profiled in different components. The components with more stems are associated with higher weights in determining the overall conformation of a sequence segment. In order to efficiently perform alignments of the sequence segment to the model, our searching algorithm aligns it to each component independently following the descending order of component weights. The statistical log-odds scores are computed based on the structural alignment scores on each CM component. Due to the conformational constraints inherently imposed by the CM components, stem contentions occur infrequently (less than 30%) and can be effectively resolved based on the conformational constraints from the alignment results on components with higher weight values. The algorithm is able to accomplish the search with a worst case time complexity of $O((k-1)W^3L)$ and a space complexity of $O(kW^2)$, where k is the number of CM components in the model, W and L are the size of the searching window and the length of the genome respectively.

We used the model to search for a variety of RNA pseudoknots inserted in randomly generated sequences. Experiments show that the model can achieve excellent sensitivity (SE) and specificity (SP) on almost all of them, while using only slightly more computation time than searching for pseudoknot-free RNA structures. We then applied the model and the searching algorithm to identify the pseudoknots on the 3' untranslated region in several RNA genomes from the corona virus family. An exact match between the locations found by our program and the real locations is observed. Finally, in order to test the ability of our program to cope with noncoding RNA genes with complex pseudoknot structures, two DNA genomes of bacterias were searched to find the location of the tmRNA genes. The results show that our program identified the locations with a small amount of error (with a right shift of around 20 nucleotide bases). To the best of our knowledge, these are the first experiments where a whole genome of more than a million nucleotides is successfully searched for a complex structure that contains pseudoknots.

2 Experiments and Results

To test the performance of the model, we developed a searching program in C language and carried out searching experiments on a Sun/Solaris workstation. The workstation has 8 dual processors and 32GB main memory. We evaluated the accuracy of the program on both real genomes and randomly generated sequences with a number of RNA pseudoknot structures inserted. To test the model, we chose 10 RNAs from 9 RNA families, tmRNA, srpRNA, Telomerase−vert, Corona−pk3, HDV−ribozyme, Tombus−3−IV, Alpha−RBS, Antizyme−FSE and IFN−gamma, which have pseudoknot annotations in Rfam [8].

Model training and testing are based on the multiple alignments downloaded from the Rfam database. For each RNA pseudoknot, we divide the available data into a training set and a testing set, and the parameters used to model it are estimated based on multiple structural alignments among $5 - 90$ homologous training sequences with pairwise identity less than 80%. Pseudocounts dependent on the number of training sequences are included to prevent overfitting of the model to the training data.

To measure the sensitivity and specificity of the searching program within a reasonable amount of time, for each selected pseudoknot structure, we selected $10 - 40$ sequence segments from the set of testing data and inserted them into each of the randomly generated sequences of 10^5 nucleotides. In order to test whether the model is sensitive to the base composition of the background sequence, we varied the C+G concentration in the random background. The program computes the log-odds, the logarithmic ratio of the probability of generating sequence segment s by the null (random) model R to that by our model M. It reports a hit when the Z-score of s is greater than 4.0.

The program correctly identifies more than 80% of inserted sequence segments with excellent specificity in most of the experiments. The only exception is the srpRNA, where the program misses more than 50% inserted sequence segments in one of the experiments. The relatively lower sensitivity in that particular experiment can be partly ascribed to the fact that the pseudoknot structure of srpRNA contains fewer nucleotides; thus, its structural and sequence patterns have larger probability to occur randomly. The running time for srpRNA, however, is also significantly shorter than that needed by most of other RNA pseudoknots due to the smaller size of the model. Additionally, while alpha−RBS pseudoknot has a more complex structure and three CM components are needed to model it, our searching algorithm efficiently identifies more than 95% of the inserted pseudoknots with high specificities. Our results demonstrate that higher C+G concentration in the background does not adversely affect the specificity of the model. The program achieves better overall performance in both sensitivity and specificity on the background of higher C+G concentrations. We therefore conjecture that the specificity of the model is partly determined by the base composition of the genome and can be improved if the base composition of the target gene is considerably different from its background.

To test the accuracy of the program on real genomes, we performed experiments to search for particular pseudoknot structures in the genomes for a variety of organisms. Table 1 shows the genomes on which we have searched with our program and the locations annotated for the corresponding pseudoknot structures. It is evident from the results that the program successfully identified the exact locations of known 3'UTR

Table 1. The results obtained with our searching program on the genomes of a variety of organisms. GA is the accession number of the genome; RL specifies the real location of the pseudoknot structure in the genome; SL is the one returned by the program; RT is the running time needed to perform the searching in hours; GL is the length of the genome in its number of bases. The genome of Haemophilus searched in our experiment is the reversed complementary DNA strand

GA	Organism	ncRNA	RL	SL	RT(hr)	GL(bs)
NC000907	Haemophilus	tmRNA	$472210 - 472575$	$472177 - 472542$	170.00	1.83×10^6
NC003112	Neisseria meningitidis	tmRNA	$1241197-$ 1241559	$1241197-$ 1241559	170.00	2.2×10^6
NC003045	Bovine CoronaVirus	3'UTR pk	$30798 - 30859$	$30798 - 30859$	1.24	31028
NC002645	Human CoronaVirus	3'UTR pk	$27063 - 27125$	$27063 - 27125$	1.12	27317
NC001846	Murine HepatitusVirus	3'UTR pk	$31092 - 31153$	$31092 - 31153$	1.27	31357
NC003436	Porcine DiarrheaVirus	3'UTR pk	$27820 - 27882$	$27820 - 27882$	1.17	28033

pseudoknot in the four genomes from the family of corona virus. This pseudoknot was recently shown to be essential for the replication of the viruses in the family [7]. In addition, we performed an experiment where the genomes of bacterias, *Haemophilus influenzae* and *Neisseria meningitidis MC58*, were searched for their tmRNA genes. The *Haemophilus influenzae* DNA genome contains about 1.8×10^6 nucleotides and *Neisseria meningitidis MC58* DNA genome contains about 2.2×10^6 nucleotides. The tmRNA functions importantly in the trans-translation process to add a C-terminal peptide tag to the incomplete protein product of a broken mRNA [12]. The central part of the secondary structure of tmRNA molecule consists of four pseudoknot structures. Figure 1 shows the pseudoknot structures on the tmRNA gene.

In order to search the DNA genomes efficiently, the combined pseudoknots 1 and 2 were used to search the genome first and the program searches for the whole tmRNA gene only in the region around the locations where a hit for Pk1 and Pk2 is detected. We cut the genome into segments with shorter lengths (around 10^5 nucleotide bases for each), and run the program in parallel on ten of them in two rounds. The result for *Neisseria meningitidis MC58* shows that we successfully identified the exact locations of tmRNA. However, the locations of tmRNA obtained for *Haemophilus influenzae* have a shift of around 20 nucleotides with respect to its real location. This error can probably be ascribed to our "hit-and-extend" searching strategy to resolve the difficulty arising from the complex structure and the relatively much larger size of tmRNA genes; positional errors may occur during different searching stages and accumulate to a significant value. Our experiment on the DNA genomes also demonstrates that, for each genome, it is very likely there is only one tmRNA gene in it, since our program found only one significant hit. To our knowledge, this is the first experiment where a whole genome of more than a million nucleotides is successfully searched for a complex structure that contains pseudoknot structures.

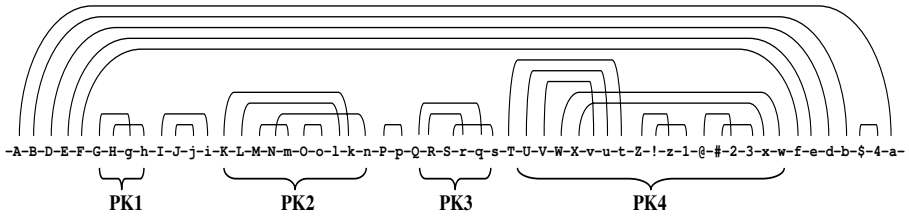


Fig. 1. Diagram of the pairing regions on the tmRNA gene. Upper case letters indicate base sequences that pair with the corresponding lower case letters. The four pseudoknots constitute the central part of the tmRNA gene and are called Pk1, Pk2, Pk3, Pk4 respectively

3 Models and Algorithms

The Covariance Model (CM) proposed by Eddy and Durbin [5][4] can effectively model the base pairs formed between nucleotides in an RNA molecule. Similar to the emission probabilities in HMMs, the emission probabilities in the CM for both unpaired nucleotides and base pairs are position dependent. The profiling of a stem hence consists of a chain of consecutive emissions of base pairs. Parallel stems on the RNA sequence are modeled with bifurcation transitions where a bifurcation state is split into two states. The parallel stems are then generated from the transitions starting with the two resulting states respectively.

The genome is scanned through by a window with an appropriate length. The ending location of the window is scored by aligning all subsequence segments contained in the window to the model with the CYK algorithm. The maximum log-odds score of the segments is determined as the log-odds score associated with the location. A hit is reported for a location if the computed log-odds score is higher than a predetermined threshold value.

Pseudoknot structures are beyond the profiling capability of a single CM due to the inherent context sensitivity of pseudoknots. Models for pseudoknot structures require a mechanism for the description of their interleaving stems. Previous work by Brown and Wilson [2] and Cai *et al.* [3] has modeled the pseudoknot structures with grammar components that intersect or cooperatively communicate. A similar idea is adopted in this work; a number of independent CM components are combined to resolve the difficulty in profiling that arises from the interleaving stems. Interleaving stems are profiled in different CM components and the alignment score of a sequence segment is determined based on a combination of the alignment scores on all components.

However, the optimal conformations from the alignments on different components may violate some of the conformational constraints that a single RNA sequence must follow. For example, a nucleotide rarely forms two different base pairs simultaneously with other nucleotides in an RNA molecule. This type of restriction is not considered by the independent alignments carried out in our model and thus may lead to erroneous searching results if not treated properly. In our model, *stem contention* may occur such that two or more base pairs obtained from different components require the participation of the same nucleotide. We break the contention by introducing different priorities to components; base pairs determined from components with the highest priority win

the contention. We consider that, biochemically, components profiling more stems are likely to play more dominant roles in the formation of the conformation and are hence assigned higher priority weights.

3.1 Model Generation

In order to profile the interleaving stems in a pseudoknot structure with independent CM components, we need an algorithm that can partition the set of stems on the RNA sequence into a number of sets comprised of stems that mutually do not interleave. Based on the consensus structure of the RNA sequence, an undirected graph $G = (V, E)$ can be constructed where V, the set of vertices in G, consists of all stems on the sequence. Two vertices are connected with an edge in G if the corresponding stems are in parallel or nested. The set of vertices V needs to be partitioned into subsets such that the subgraph induced by each subset forms a clique.

We use a greedy algorithm to perform the partition. Starting with a vertex set S initialized to contain a arbitrarily selected vertex, the algorithm iteratively searches the neighbors of the vertices in S and computes the set of vertices that are connected to all vertices in S. It then randomly selects one vertex v that is not in S from the set and modifies S by assigning v to S. The algorithm outputs S as one of the subsets in the partition when S can not be enlarged and randomly selects an unassigned vertex and repeats the same procedure. It stops when every vertex in G has been included in a subset. Although the algorithm does not minimize the number of subsets in the partition, our experiments show that it can efficiently provide optimal partitions of the stems on pseudoknot structures of moderate structural complexity.

The CM components in the profiling model are generated and trained based on the partition of the stems. The stems in the same subset are profiled in the same CM component. For each component, the parameters are estimated by considering the consensus structure formed by the stems in the subset only.

3.2 Searching Algorithm

The optimal alignments of a sequence segment to the CM components are computed with the dynamic programming based CYK algorithm. As we have mentioned before, higher priority weights are assigned to components with more stems profiled. The component with the maximum number of stems thus has the maximum weight and is the *dominant component* in the model. The algorithm performs alignments in the descending order of component weights. It selects the sequence segment that maximizes the log-odds score from the dominant component. The alignment scores and optimal conformations of this segment on other components are then computed and combined to obtain the overall log-odds score for its corresponding position on the genome.

More specifically, we assume that the model contains,in descending order of component weights, k CM components $M_0, M_1, \cdots, M_{k-1}$. The algorithm considers all possible sequence segments s_d that are enclosed in the window and uses Equation (1) to determine the sequence segment s to be the candidate for further consideration, where W is the length of the window used in searching, and Equation (2) to compute the overall log-odds score for s. We use sm_i to denote the parts of s that are aligned to the stems profiled in CM component M_i. Basically, $Log_odds(sm_i|M_i)$ accounts for the

contributions from the alignment of sm_i to M_i. The log-odds score of sm_i is counted in both M_0 and M_i and must be subtracted from the sum.

$$s = \arg \max_{0<|s_d|<W} \{Log_odds(s_d|M_0)\}. \quad (1)$$

$$Log_odds(s|M) = Log_odds(s|M_0) + \sum_{i=1}^{k-1} \sum_{sm_i \in M_i} (Log_odds(sm_i|M_i) - Log_odds(sm_i|M_0)). \quad (2)$$

4 Conclusions and Future Work

In this paper, we have introduced a new model that serves as the basis for a generic framework that can efficiently search genomes for the noncoding RNAs with pseudoknot structures. Within the framework, interleaving stems in pseudoknot structures are modeled with independent CM components and alignment is performed by aligning sequence segments to all components following the descending order of their weight values. Stem contention occurs with a low frequency and can be resolved with dynamic programming based recomputation. The statistical log-odds scores are computed based on the alignment results from all components. Our experiments on both random and biological data demonstrate that the searching framework achieves excellent performance in both accuracy and efficiency and can be used to annotate genomes for noncoding RNA genes with complex secondary structures in practice.

We were able to search a bacterial genome in about one week on our Sun workstation. It would be desirable to improve our algorithm so that we could search larger genomes and databases. The running time, however, could be significantly shortened if a filter can be designed to preprocess DNA genomes and only the parts that pass the filtering process are aligned to the model. Alternatively, it may be possible to devise alternative profiling methods to the covariance model that would allow faster searches.

Supplementary Material

We have put RNA information we used for the estimation of model parameters, the experimental results of the model on the RNA pseudoknots, and the detailed stem contention explanation and experimental results for stem contention rates on the web. They are available at http://www.uga.edu/RNA-Informatics/pksearch/.

References

1. T. Akutsu: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104: 45-62, 2000
2. M. Brown and C. Wilson: RNA Pseudoknot Modeling Using Intersections of Stochastic Context Free Grammars with Applications to Database Search. *Pacific Symposium on Biocomputing*, 109-125, 1995

3. L. Cai, R. Malmberg, and Y. Wu: Stochastic Modeling of Pseudoknot Structures: A Grammatical Approach. *Bioinformatics*, 19, $i66 - i73$, 2003
4. R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. *Cambridge University Press*, 1998
5. S. Eddy and R. Durbin: RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22: 2079-2088, 1994
6. D. Gautheret and A. Lambert: Direct RNA motif definition and identification from multiple sequence alignments using secondary structure profiles. *Journal of Molecular Biology*, 313: 1003-1011, 2001
7. S. J. Geobel, B. Hsue, T. F. Dombrowski, and P. S. Masters: Characterization of the RNA components of a Putative Molecular Switch in the 3' Untranslated Region of the Murine Coronavirus Genome. *Journal of Virology*, 78: 669-682, 2004
8. S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy: Rfam: an RNA family database. *Nucleic Acids Research*, 31: 439-441, 2003
9. A. Krogh, M. Brown, IS. Mian, K. Sjolander, and D. Haussler: Hidden Markov models in computational biology. Applications to protein modeling. *Journal of Molecular Biology*, 235: 1501-1531, 1994
10. R. B. Lyngso and C. N. S. Pederson: RNA pseudoknot prediction in energy based models. *Journal of Computational Biology*, 7: 409-428, 2000
11. T. Macke, D. Ecker, R. Gutell, and D. Gautheret, D. Case, R. Sampath: RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Research*, 29: 4724-4735, 2001
12. N. Nameki, B. Felden, J. F. Atkins, R. F. Gesteland, H. Himeno, A. Muto: Functional and structural analysis of a pseudoknot upstream of the tag-encoded sequence in E. coli tmRNA. *Journal of Molecular Biology*, 286(3): 733-744, 1999
13. E. Rivas and S. Eddy: The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics*, 16: 334-340, 2000
14. E. Rivas and S. Eddy: A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots. *Journal of Molecular Biology*, 285: 2053-2068, 1999
15. J. Ruan, G. D. Stormo, and W. Zhang: An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics*, 20: 58-66, 2004
16. Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjolander, R. C. Underwood, and D. Maussler: Stochastic Context-Free Grammars for tRNA Modeling. *Nucleic Acids Research*, 22: 5112-5120, 1994

Integrating Text Chunking with Mixture Hidden Markov Models for Effective Biomedical Information Extraction

Min Song, Il-Yeol Song, Xiaohua Hu, and Robert B. Allen

College of Information Science & Technology Drexel University,
Philadelphia, PA 19104
(215) 895-2474, 01
{min.song, song, thu, rba}@drexel.edu

Abstract. This paper presents a new information extraction (IE) technique, KXtractor, which integrates a text chunking technique with Mixture Hidden Markov Models (MiHMM). KXtractor is differentiated from other approaches in that (a) it overcomes the problem of the single Part-Of-Speech (POS) HMMs with modeling the rich representation of text where features overlap among state units such as word, line, sentence, and paragraph. By incorporating sentence structures into the learned models, KXtractor provides better extraction accuracy than the single POS HMMs do. (b) It resolves the issues with the traditional HMMs for IE that operate only on the semi-structured data such as HTML documents and other text sources in which language grammar does not play a pivotal role. We compared KXtractor with three IE techniques: 1) RAPIER, an inductive learning-based machine learning system, 2) a Dictionary-based extraction system, and 3) single POS HMM. Our experiments showed that KXtractor outperforms these three IE systems in extracting protein-protein interactions. In our experiments, F-measure for KXtractor was higher than ones for RAPIER, a dictionary-based system, and single POS HMM respectively by 16.89%, 16.28%, and 8.58%. In addition, both precision and recall of KXtractor are higher than those systems.

1 Introduction

The proliferation of the biomedical literature available on the Web is overwhelming. While the amount of data available to us is constantly increasing, our ability to absorb and process this information remains a challenging task. The biomedical research domain has recently become a target domain that information extraction (IE) can be spotlighted on. IE is the process of scanning text for information relevant to some interest, including extracting entities, relations, and events. In this paper, we propose a novel IE technique, called KXtractor, Mixture Hidden Markov Models (MiHMMs) combined with a Support Vector Machine (SVM)-based text chunking technique.

MiHMM is defined as a mixture of Hidden Markov Models (HMMs) organized in a hierarchical structure to help the IE system cope with data sparseness. MiHMM takes a set of sentences with contextual cues that were identified by a Support Vector Machine-based text chunking technique. MiHMM then learns a generative

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 976 – 984, 2005.
© Springer-Verlag Berlin Heidelberg 2005

probabilistic model of the underlying state transition structure of the sentence from a set of tagged training data. Given a trained probabilistic mixture model of the data, the system is able to apply this model to new unseen input documents to predict which portions of these sentences are likely targets according to the training data template. The paper investigates relationships between structure and performance of HMMs applied to information extraction problems.

KXtractor is different from existing HMMs as follows: (a) It employs probabilistic mixture of HMMs that are hierarchically structured. (b) It incorporates contextual and semantic cues into the learned models to extract knowledge from the unstructured text collections without any document structures. (c) It adopts an SVM text chunking technique to partition sentences into grammatically related groups. Thus, using KXtractor for extracting biomedical entities has the following advantages over other approaches: (a) it overcomes the problem of the single POS HMMs with modeling the rich representation of text where features overlap among state units such as word, line, sentence, and paragraph. By incorporating sentence structures into the learned models, KXtractor provides better extraction accuracy than the single POS HMM does. (b) it resolves the issues with the single POS HMMs for IE that operate only on the semi-structured data such as HTML documents and other text sources in which language grammar does not play a pivotal role.

With this novel and robust IE technique, we have extracted protein-protein pairs from abstracts in MEDLINE. We have compared the system performance of KXtractor with other IE techniques such as a rule-based learning, a dictionary-based, and single POS HMM techniques. Our experimental results show that KXtractor is superior to these techniques in most cases. The rest of the paper is organized as follows: Section 2 describes the overall architecture of KXtractor. Section 3 describes the evaluation. Section 4 reports on the experiments. Section 5 concludes the paper.

2 The System Architecture

Figure 1 illustrates the system architecture of KXtractor. The system consists of two major components: 1) sentence chunking by an SVM component and 2) relation extraction by MiHMM component.

In the sentence chunking by an SVM component, input data is plain text consisting of titles and abstracts. The input data is separated into sentences. A set of regular expression rules are applied to parse sentences. For a parsed sentence, we applied an integrated POS tagging technique proposed by Song et al. [9] to tag sentences with POS. With SVM-based text chunking technique, these POS tagged sentences are then grouped into chunks of different phrase types such as noun, verb, and preposition chunk. In the relation extraction by MiHMM component, MiHMM is applied to the grouped phrases by the SVM text chunking technique. The results of running KXtractor are a set of tuples related to protein-protein pairs. KXtractor stores these tuples in the knowledge base and resets the token statistics for the next input data. The detailed description of the components is provided in the sub-sections below. Figure 2

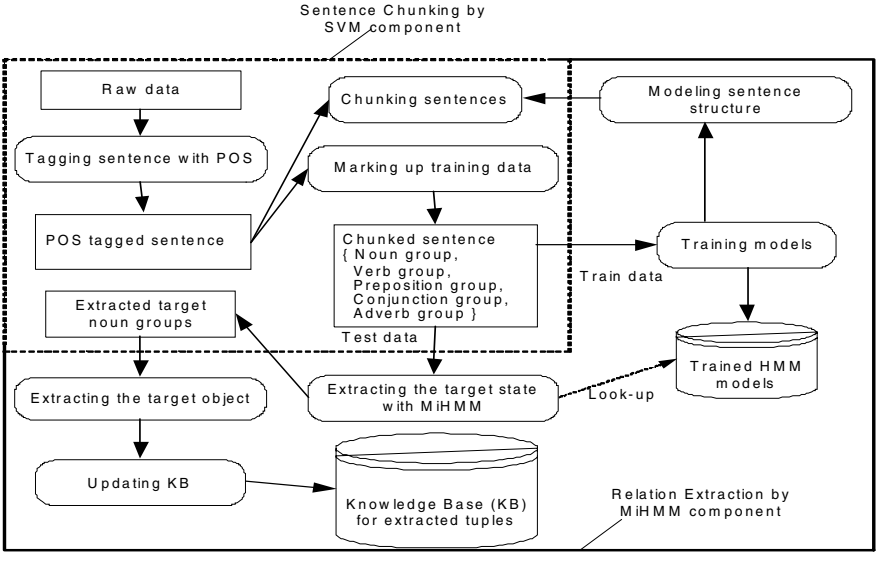


Fig. 1. System architecture of KXtractor

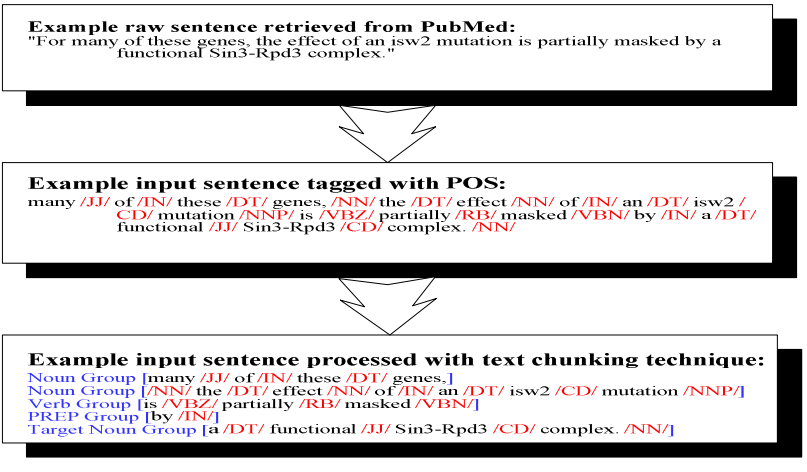


Fig. 2. A procedure of sentence parsing JJ denotes adjective, IN denotes preposition, DT denotes determiner, CD cardinal number, NN denotes singular noun, NNP denotes proper noun, VBZ denotes verb, VBN denotes verb, and RB denotes adverb

illustrates the procedure of converting a raw sentence from PubMed to the phrase-based units grouped by the SVM text chunking technique. The top box shows a sentence that is part of abstracts retrieved from PubMed. The middle box illustrates

the parsed sentence by POS taggers. The bottom box shows the final conversion made to the POS tagged sentence by the SVM based text chunking technique.

2.1 Sentence Chunking by an SVM Component

Text chunking is defined as dividing a text in syntactically correlated parts of words [5]. Chunking is recognized as a series of the processes - first identifying proper chunks from a sequence of tokens (such as words), and second classifying these chunks into some grammatical classes. Major advantages of using text chunking over full parsing techniques are that partial parsing such as text chunking is much faster, more robust, yet sufficient for IE. Text chunking based on Support Vector Machines (SVMs) was reported to produce the highest accuracy in the text chunking task [5]. The SVMs-based approaches such as other inductive-learning approaches take as input a set of training examples (given as binary valued feature vectors) and find a classification function that maps them to a class. SVMs are known to robustly handle large feature sets and to develop models that maximize their generalizability. This makes them an ideal model for IE.

Generalizability in SVMs is based on statistical learning theory and on the observation that it is useful to misclassify some of the training data so that the margin between other training points is maximized [4]. This is particularly useful for real world data sets that often contain inseparable data points. Although training is generally slow, the resulting model is usually small and runs quickly. This is because only the patterns that help define the function that separates positive from negative examples. In addition, SVMs are binary classifiers and so we need to combine SVM models to obtain a multiclass classifier. Due to the nature of the SVM as a binary classifier it is necessary in a multi-class task to consider the strategy for combining several classifiers. In this paper, we use Tiny SVM [4] in that Tiny SVM performs well in handling a multi-class task.

2.2 Relation Extraction by MiHMM Component

Figure 3 is a schematic representation of how our MiHMM works. Our phrase group includes 14 phrase types. Our models are constructed with the assumption that the model is fully connected, which means that the model emits a segment of any type at any given position within the sentence. Bold boxes in Figure 3 indicate the target noun group that contains either proteins or a protein-protein pair. Each box represents a phrase group and the circles inside the box show the POS tags assigned to words in order, which appears in the sentence. The model is trained maximum likelihood parameter estimation. From the sentence training set we can easily obtain the information concerning the frequency that a given state or observation occurred and the frequency with which a state transition or observation emission was made. The parameters of the model are the transition probabilities $P(q \rightarrow q')$ that one state follows another and the emission probabilities $P(q \uparrow q')$ that a state emits a particular output symbol. The probability of a string x being emitted by an HMM is computed as a sum over all possible paths by:

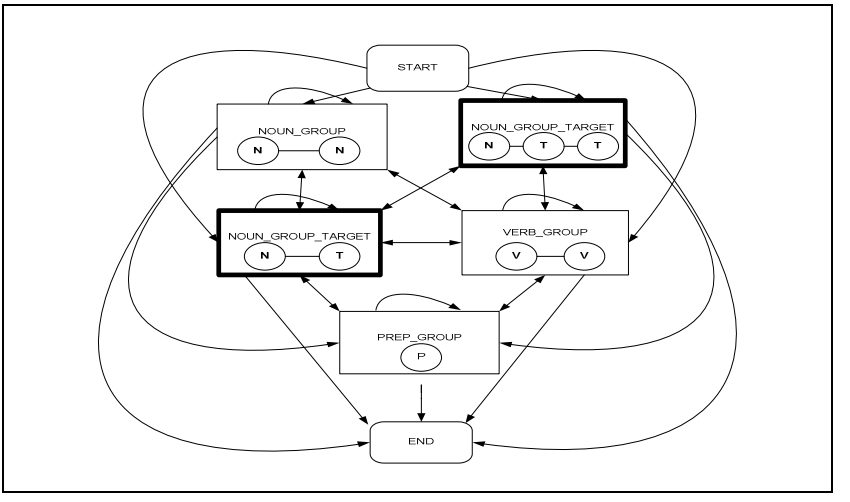


Fig. 3. Noun phrase based Mixture Hidden Markov ModelsN denote noun, P denotes preposition, T denotes target, and V denote verb

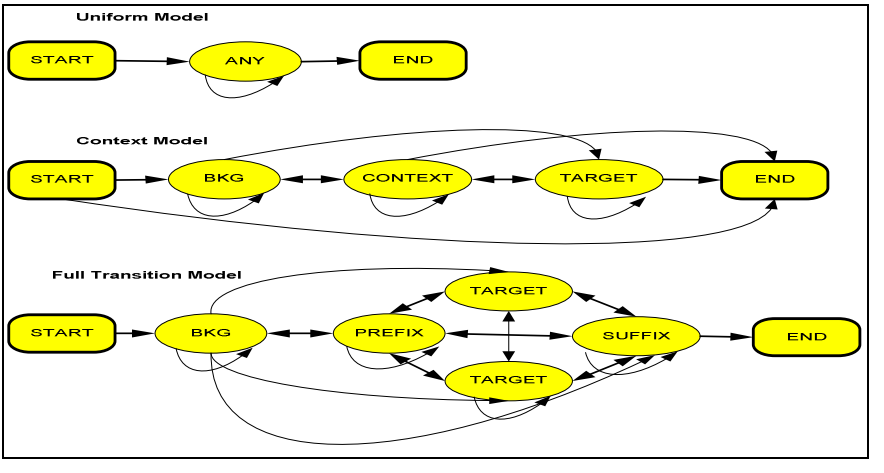


Fig. 4. Graphic representation of MiHMM. BKG denotes Background

$$P(x \mid M) = \sum_{q_1 \ldots q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k) P(q_k \uparrow x_k)$$

where q_0 and q_{l+1} are restricted to be q_I and q_F respectively, and is an end-of-string token. The forward algorithm can be used to calculate this probability [7]. The observable output of the system is the sequence of symbols that the states emit, but the underlying start sequence itself is hidden. One common goal of learning problems

that use HMMs is to recover the state sequence $V(x \mid M)$ that has the highest probability of having produced an observation sequence:

$$V(x \mid M) = \arg \max_{q_1 \ldots q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k) P(q_k \uparrow x_k)$$

Determining this state sequence is efficiently performed by dynamic programming with the Viterbi algorithm [10].

3 Evaluation

To evaluate KXtractor, we compare it with three other well-known IE methods: 1) the dictionary-based extraction, 2) RAPIER, a rule-based machine learning extraction, and 3) single POS HMM. Performance of these IE systems is measured by precision, recall, and F-measure. The data used for experiments are retrieved from MEDLINE.

3.1 Data Collection

The IE task conducted in this paper is a multiple slot extraction task. The goal of our IE task is to extract instances of *n*-ary relations; that is, protein-protein interactions. The protein-protein interaction data sets are composed of abstracts gathered from the MEDLINE database [6]. MEDLINE contains bibliographic information and abstracts from more than 4000 biomedical journals. From this huge text corpus, we combined and utilized MEDLINE data sets provided by [8] and [2]. The data sets consist of 1700 MEDLINE records. These data sets characterize physical interactions between pairs of proteins. In terms of sentences, the data sets consist of 6417 positive and 46123 negative sentences. It contains 10123 instances of 913 protein-protein pairs. To label the sentences in these abstracts, we matched the target tuples to the words in the sentence. A sentence that contained words that matched a tuple was taken to be a positive instance. Every other sentence was considered to be a negative instance.

3.2 Dictionary-Based Extraction

We developed a Dictionary-based extraction system proposed by Blaschke et al. As described in [2], the following six steps were taken to extract protein-protein interactions: 1) the protein names are collected from the Database of Interacting Proteins (DIP) and Protein-Protein Interaction Database (PPID) databases. The synonyms of the target proteins are manually provided. 2) The 14 verbs, indicating actions related to protein interaction, are used. 3) Abstracts are provided from MEDLINE. 4) The passages containing target proteins and actions are identified. 5) The original text is parsed into fragments preceding grammatical separators. 6) The final step is to build protein-protein pairs.

3.3 RAPIER

To evaluate the performance of KXtractor, we compare KXtractor with RAPIER. RAPIER [3] is a well-known IE system that was developed with a bottom-up

inductive learning technique for learning information extraction rules. In order to use the slot-filling IE systems like RAPIER for extracting relations, we adapt the Role-filler approach proposed by Bunescu et al. [2]. The Role-filler approach allows for extracting the two related entities into different role-specific slots. For protein interactions, Bunescu et al. [2] name the roles interactor and interactee. As indicated by the role names, protein-protein interactions are defined with the assumption that proteins appear in the same sentence.

3.4 Single POS HMM

In order to verify that our MiHMM models are superior to the plain HMM, we develop a plain HMM, based on single terms and a single model that incorporate less grammatical information. We implemented single-level HMMs whose states emit words, but are typed with part-of-speech (POS) tags so that a give state can emit words with only a single POS. The Viterbi algorithm extracts information from documents modeled by an HMM. With the fix structure, the objective of learning is to give high probabilities to training documents. The result of learning is estimated probabilities for vocabularies and transitions.

4 Experiments

We conducted experiments to evaluate the performance of KXtractor on the task of protein-protein interaction extraction. In experiments the machine learning systems were trained using the abstracts with proteins and their interactions, processed by the text chunking technique. With these set of data, the IE systems extract interactions among these proteins. This gives us a measure of how the extraction systems for protein-protein pairs perform alone. Performance is evaluated using ten-fold cross validation and measuring recall and precision. As the task of interest is only to extract interacting protein-pairs, in our evaluation we do not consider matching exact position and every occurrence of interacting protein-pairs within the abstract.To evaluate our IE systems, we construct a precision-recall graph. Recall denotes the ratio of the number of slots the system found correctly to the number of slots in the answer key, and precision is the ratio of the number of correctly filled slots to the total number of slots the system filled.

Our experiments show that RAPIER produces relatively high precision but low recall. The similar results are observed in the dictionary-based extraction method which gives also high precision but low recall. Single POS HMM produces the second best results, although recall is relatively lower than precision. Among these three systems, KXtractor outperforms RAPIER, Dictionary, and single POS HMM in terms of precision, recall, and F-measure. As shown in Table 1, F-Measure of KXtractor is 52.38% whereas RAPER is 35.49%, dictionary is 36.10%, and single POS HMM is 43.80%. Figure 5 shows the precision-recall graphs of KXtractor, RAPIER, Dictionary, and single POS HMM-based extraction for the protein-protein interaction data set. The curve for KXtractor is superior to the curves for RAPIER, Dictionary, and single POS HMM.

Table 1. Comparison of extraction system performance

Extraction System	*Precision*	*Recall*	*F-Measure*
Dictionary-based extraction	62.31%	32.81%	36.10%
RAPIER	60.17%	34.12%	35.49%
Single POS HMM	67.40%	47.23%	43.80%
KXtractor	70.23%	51.21%	52.38%

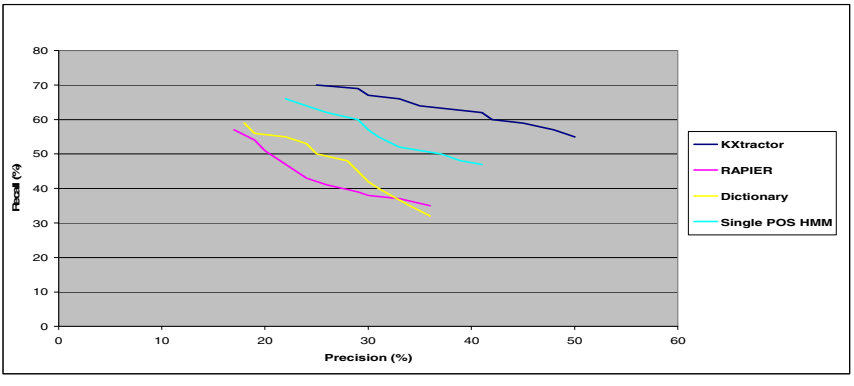


Fig. 5. Precision-recall graph for extracting protein-protein pairs

5 Conclusion

In this paper, we proposed a novel and high quality information extraction system, called KXtractor, a noun phrase-based Mixture Hidden Markov Models (MiHMM) system. KXtractor consists of two major components: 1) text chunking and 2) Mixture Hidden Markov Models (MiHMM) component. KXtractor is differentiated from other approaches in that (a) It overcomes the problem of the single POS HMMs with modeling the rich representation of text where features overlap among state units such as word, line, sentence, and paragraph. By incorporating sentence structures into the learned models, KXtractor provides better extraction accuracy than the single POS HMMs. (b) It resolves the issues with the single POS HMMs for IE that operate only on the semi-structured data such as HTML documents and other text sources in which language grammar does not play a pivotal role.

We compared KXtractor with three well-known IE techniques: 1) RAPIER, a rule-based machine learning system, 2) Dictionary-based extraction system which was proposed by [2], and 3) single POS HMM. Our experiments showed that KXtractor outperforms other IE techniques such as RAPIER, dictionary-based, and single POS HMM in extracting protein-protein interactions in terms of F-measure. The F-

Measure of KXtractor is 52.38% whereas RAPER is 35.49%, dictionary is 36.10%, and single POS HMM is 43.80%. In addition, both precision and recall of KXtractor are higher than those of RAPIER, Dictionary, and single POS HMM. In our follow-up paper, we will apply KXtractor to other types of relation extractions such as subcellular-localization relation extraction. We also plan to compare KXtractor with other IE systems such as MaxEnt and SVM.

References

1. Blaschke, C., Andrade, M.A., Ouzounis, C., and Valencia, A. (1999). Automatic Extraction of Biological Information from Scientific Text: Protein-Protein Interactions, *In Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology,* Heidelberg, Germany, 60-67.
2. Bunescu, R., Ge, R., Kate, R.J., Marcotte, E.M., Mooney, R.J., Ramani, A.K., and Wong, Y.W. (2004). Comparative Experiments on Learning Information Extractors for Proteins and their Interactions. To appear in *Journal Artificial Intelligence in Medicine (Special Issue on Summarization and Information Extraction from Medical Documents).*
3. Califf, M.E. and Mooney, R.J. (1999). Relational Learning of Pattern-Match Rules for Information Extraction. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, FL, 328-334.
4. Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3): 273-297.
5. Kudo, T. and Matsumoto, Y. (2000). Use of Support Vector Learning for Chunk Identification. In Proceedings of CoNLL- 2000 and LLL-2000, Saarbruncken, Germany, 142-144.
6. National Library of Medicine (2003). *The MEDLINE database*, http://www.mcbi.nlm.nih.gov/PubMed/.
7. Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257-286.
8. Skounakis, M., Craven, M., and Ray, S. (2003). Hierarchical Hidden Markov Models for Information Extraction. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, August, 427-433.
9. Song, M, Song, I-Y., and Hu, X. (2003). KPSpotter: A Flexible Information Gain-based Keyphrase Extraction System, *Fifth International Workshop on Web Information and Data Management (WIDM'03)*, New Orleans, Lousiana, 50-53.
10. Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Processing*, 13:260-269.

k-Recombination Haplotype Inference in Pedigrees

Francis Y.L. Chin[1], Qiangfeng Zhang[2], and Hong Shen[3]

[1] Department of Computer Science,
The University of Hong Kong, Pokfulam, Hong Kong
chin@cs.hku.hk
[2] Department of Computer Science,
University of Science and Technology of China, Hefei, China
qfzhang@mail.ustc.edu.cn
[3] Graduate School of Information Science, JAIST, Ishikawa, Japan
shen@jaist.ac.jp

Abstract. Haplotyping under the Mendelian law of inheritance on pedigree genotype data is studied. Because genetic recombinations are rare, research has focused on Minimum Recombination Haplotype Inference (MRHI), i.e. finding the haplotype configuration consistent with the genotype data having the minimum number of recombinations. We focus here on the more realistic k-MRHI, which has the additional constraint that the number of recombinations on each parent-offspring pair is at most k.

Although k-MRHI is NP-hard even for k = 1, we give an algorithm to solve k-MRHI efficiently by dynamic programming in O(nm03k+12m0) time on pedigrees with n nodes and at most m0 heterozygous loci in each node. Experiments on real and simulated data show that, in most cases, our algorithm gives the same haplotyping results but runs much faster than other popular algorithms.

1 Introduction

The modeling of human genetic variation is critical to the understanding of the genetic basis for complex diseases. *Single nucleotide polymorphisms* (*SNPs* [13]) are the most frequent form of variation. The Human Genome Project and other large-scale efforts have identified millions of SNP markers. Although each marker can be analyzed independently, it is more informative to analyze them in groups. Therefore, it is useful to analyze *haplotypes* (*hap*loid geno*types*), which are sequences of linked markers on a single chromosome. In diploid organisms, such as humans, chromosomes come in pairs, and experiments often yield *genotype* information, which blends haplotype information for chromosome pairs. Given that deriving haplotype information experimentally is time-consuming and expensive, there is increasing interest in inferring haplotype information, or *haplotyping*, computationally [4][7] from genotype information.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 985 – 993, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Haplotyping *pedigree* data is believed to be more reliable than haplotyping population data for unrelated individuals, and genetic research shows that recombinations are rare in human pedigree data [6]. The *Minimum-Recombination Haplotype Inference* (*MRHI*) problem, which is NP-hard [5], is to find a haplotype configuration with minimum number of recombinations given pedigree genotype data [8][9][12][15]. At times, however, the MRHI model might yield unrealistic results in which recombinations are concentrated in a few parent-offspring pairs. The k-MRHI problem is basically MRHI but with the additional constraint that the number of recombinations in each parent-offspring pair is bounded by a constant k.

The k-MRHI problem is NP-hard even for $k = 1$, but can be solved by a dynamic programming (DP) algorithm similar to the algorithm in [5] for MRHI. By avoiding studying all 2^{3m_0} haplotype configurations in each parents-offspring trio, our algorithm takes $O(nm_0{}^4 2^{m_0})$ time when $k = 1$, instead of $O(nm_0 2^{3m_0})$ [5] for the MRHI problem on pedigrees with n nodes and at most m_0 heterozygous loci in each node. Not all nodes have m_0 heterozygous loci, and the number of feasible haplotype configurations at a node is limited by that of its neighbors and thus can be much less than 2^{m_0}. This observation leads to the idea of choosing a good root node to speed up the algorithm.

This paper's main contributions are: (1) to define a more realistic problem for haplotype inference (k-MRHI), (2) to give a more efficient and practical DP algorithm for k-MRHI, and (3) to present an efficient algorithm to find a good root in the pedigree to improve the DP algorithm's performance.

2 Preliminaries

Haplotypes and genotypes consist of linked *genetic markers* which are small DNA segments. The physical position of a marker on a chromosome is called a *locus* and its state is called an *allele*. The two alleles of a biallelic (2-state) SNP can be denoted by '0' and '1', and a haplotype h with m loci is presented as a string of length m over $\{0,1\}^m$, and a genotype g as a string over $\{0,1,2\}^m$. Haplotype pair $\langle h_1, h_2 \rangle$ is *consistent* with genotype g if (a) the two alleles of h_1 and h_2 are the same at the same locus (a *homozygous* site), for example '0' (respectively '1'), then the corresponding locus of g should also be '0' (respectively '1'); otherwise, (b) the two alleles of h_1 and h_2 are different, then the corresponding site of g should be '2' (a *heterozygous* site).

Figure 1(a) shows the pictorial representation of a pedigree with 13 nodes. A square represents a male node, a circle a female node, and a black dot a mating node. A **pedigree** can be formally defined as a weakly connected directed a cyclic graph $P = (V, E)$, where $V = M \cup F \cup N$, with M stands for the male nodes, F the female nodes, N the mating nodes, and $E = \{(u, v) | u \in M \cup F \text{ and } v \in N, \text{ or alternatively } u \in N \text{ and } v \in M \cup F\}$.

Figure 1(b) shows the graph representation of the pedigree given in Figure 1(a). A *nuclear family* comprises a father, a mother, and their children (see subgraph in

the dotted square) and can also be represented by a mating node which connects them together. A *parents-offspring trio*, or just *trio*, consists of two parents and one of their children, and a *parent-offspring pair* (*PO-pair*) refers to a father and his child or a mother and her child. In this paper, we assume that the pedigree never forms a cycle if the directions of edges are ignored (no *mating-loop*).

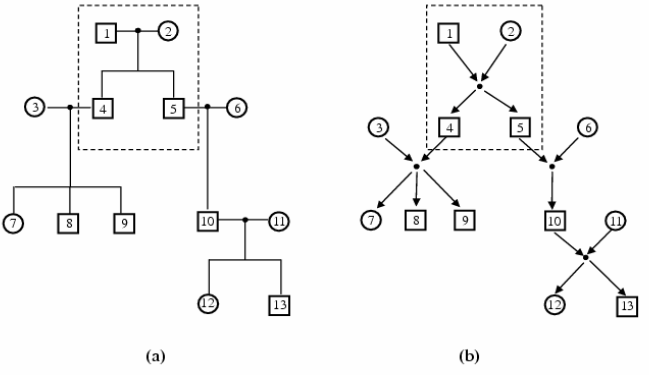


Fig. 1. The pictorial representation and graph representation of a pedigree

In the absence of genetic mutation, at each locus, the child must inherit one allele from its father and the other from its mother, i.e. the *Mendelian law of inheritance*. Usually, one haplotype of a child is inherited as a whole from one of the two haplotypes of a parent. However, *recombinations* may occur, where the two haplotypes of a parent get shuffled due to a crossover of a chromosome and one of the shuffled copies (*recombinant*) is passed on to the child. Genetic research shows that recombinations are rare in human genetics. The *Minimum Recombinant Haplotype Inference* (*MRHI*) problem [5] is to find the haplotype configuration such that the total number of recombinations in the whole pedigree is minimized.

Since it is rare to have many recombinations within one PO-pair, we focus on the *k-Recombination Haplotype Inference* (*k-MRHI*) problem to find a haplotype configuration that is consistent with the genotypes at all nodes having the minimum number of recombinations and no more than k recombinations in each PO-pair.

3 A Dynamic Programming Algorithm for *k*-MRHI

3.1 The 1-MRHI Problem ($k = 1$)

In [5], Doi *et al.* gives a proof for the NP-hardness of the MRHI problem, which trivially implies that the k-MRHI problem, even for $k = 1$, is also NP-hard. We shall focus on the 1-MRHI problem first and generalize to the k-MRHI problem later.

We adopt a locus-based dynamic programming (DP) approach to solve the 1-MRHI problem by assigning an arbitrary node R in the pedigree as the root and recursively finding $num[R][s]$, where $num[r][s]$ denotes the minimum number of recombinations required in the sub-tree rooted at r with the haplotype configuration s under the constraint that there is at most 1 recombination in each PO-pair of the sub-tree. If r has multiple mating nodes as its tree sons, we compute each mating node separately. Each child mating node of r, comprising father F, mother M and children C_1, …, C_d. If r is a leaf node, $num[r][s] = 0$ for any haplotype configuration s; else, if r is M (or F, respectively) with haplotype configuration s, then:

$$num[r][s] = \min_{p}\{num[F][p] + \sum_{i=1}^{d} \min_{c_i}(num[C_i][c_i] + numtrio(p, s, c_i))\} \tag{1}$$

where p denotes the haplotype configuration at node F and c_i the haplotype configuration at C_i. $numtrio(p, s, c_i)$ returns the minimum number of recombinations required for a trio comprising F, M, and C_i with the haplotype configurations p, s and c_i respectively, under the constraint that no PO-pair can have more than one recombination. If there does not have any feasible solution, then $numtrio(p, s, c_i)$ returns ∞.

Similarly, if r is C_j with haplotype configuration s, then we have:

$$\begin{aligned} num[r][s] = \quad & \min_{p,q}\{numtrio(p,q,s) + num[F][p] + num[M][q] \\ & + \sum_{i=1, i\neq j}^{d} \min_{c_i}(num[C_i][c_i] + numtrio(p,q,c_i))\}, \text{ where } r = C_j \end{aligned} \tag{2}$$

Note that the above algorithm is the same as that presented in [5], but a reduction in time complexity is possible because it is not necessary to consider all combinations of haplotype configurations in each trio, which number $O(2^{3m_0})$ in total. In fact, many combinations of haplotype configurations will be infeasible, i.e. have more than one recombination per PO-pair.

For example, assume the genotype of M is (2, 2, …, 2) of length m_0 and with haplotype configuration $s = \langle h_{s1}, h_{s2}\rangle$ and h_{c1} in the haplotype $c = \langle h_{c1}, h_{c2}\rangle$ of C_i is inherited from s with no more than one recombination. There are m_0+1 ways of forming h_{c1} by inheriting its first w alleles from the first w alleles in h_{s1} and the remaining (m_0-w) alleles from h_{s2} with $0 \le w \le m_0$. Similarly, there are another m_0+1 ways of forming h_{c1} from the first w alleles in h_{s2} and the remaining (m_0-w) alleles from h_{s1}. Let N_s be the set of feasible haplotype configurations $c = \langle h_{c1}, h_{c2}\rangle$ that can be inherited by child C_i from s of r with no more than one recombination. Thus, $|N_s| \le 2(m_0+1)$. As h_{c2} is inherited from the haplotype configuration $q = \langle h_{q1}, h_{q2}\rangle$ of F, let N_c be the set of feasible haplotype configurations of F which can produce the haplotype configuration c in C with no more than one recombination. Let $N'_{s,C_i} = \cup_{c\in N_s} N_c$, which indicates the set of feasible haplotype configurations in F which can go together with haplotype s in M to produce C_i. Obviously, $N'_{s,C_i} \le 4(m_0+1)^2$.

As each haplotype configuration in F should be able to produce any of the children C_1, …, C_d, the set of feasible haplotype configurations in F is $N'_s = \cap_i N'_{s,C_i}$. Equation (1) can be rewritten as:

$$num[r][s] = \min_{p \in N'_s} \{num[F][p] + \sum_i \min_{c_i \in N_s} (num[C_i][c_i] + numtrio(p, s, c_i))\} \quad (3)$$

As for Equation (2), if r is C_j and its haplotype configuration $s = <h_{s1}, h_{s2}>$, let $N_{s,F}$ and $N_{s,M}$ be the sets of feasible haplotype configurations in F and M, which can produce C_j with haplotype configuration s. As $|N_{s,F}| \le 2(m_0+1)$ and $|N_{s,M}| \le 2(m_0+1)$, let N_{p,C_i} (N_{q,C_i}) be the set of feasible haplotype configurations on another child C_i with haplotype configuration p in F (q in M) and $N''_{p,q} = N_{p,C_i} \cap N_{q,C_i}$ be the set of feasible haplotype configurations for each child C_i which can concurrently appear with the haplotype configuration s of child C_j. Note that $N''_{p,q} \le 2(m_0+1)$ and Equation (2) can be rewritten as

$$num[r][s] = \min_{p \in N_{s,F}, q \in N_{s,M}} \{numtrio(p, q, s) + num[F][p] + num[M][q] + \sum_{i \ne j} \min_{c_i \in N''_{p,q}} (num[C_i][c_i] + numtrio(p, q, c_i))\}, \text{where } r = C_j \quad (4)$$

Theorem 1. *The 1-MRHI problem can be solved in $O(nm_0^4 2^{m_0})$ time and $O(n2^{m_0})$ space for a pedigree with n nodes and at most m_0 heterozygous loci in each node.*

Proof. Since we need $O(m_0)$ time to compute *numtrio* for each haplotype configuration combination in a trio and there are $8(m_0+1)^3$ haplotype configuration combinations in each trio, it takes $O(m_0^4 2^{m_0})$ time to process each trio. With at most n trios in the pedigree, time complexity is $O(nm_0^4 2^{m_0})$. Space complexity is $O(n2^{m_0})$ based on the size of *num*. ∎

3.2 The *k*-MRHI Problem

Here we generalize the DP algorithm for the general k-MRHI problem by modifying the definition of neighboring haplotype configurations set from N_s to $N_s^{(k)}$: the set of haplotype configuration $c = <h_{c1}, h_{c2}>$ from s with no more than k recombinations. So we have $|N_s^{(k)}| = O(m_0^k)$. Similarly, we modify the definition of $N'^{(k)}_s = \cap_i N'^{(k)}_{s,C_i}$ with $N'_{s,C}$ to $N'^{(k)}_{s,C}$ in Equation (3) and the definition of $N_{s,F}$ and $N_{s,M}$ to $N_{s,F}^{(k)}$ and $N_{s,M}^{(k)}$, N_{p,C_i} to $N_{p,C_i}^{(k)}$, and $N''_{p,q}$ to $N''^{(k)}_{p,q}$ in Equation (4).

Theorem 2. *The k-MRHI problem can be solved in $O(nm_0^{3k+1}2^{m_0})$ time for a pedigree with n nodes and at most m_0 heterozygous loci in each node.*

4 Root Selection for Better Performance

For 1-MRHI, the feasible haplotype configuration combinations in each trio may be much less than $O(m_0^3 2^{m_0})$ in practice because: (1) not all nodes have m_0

heterozygous loci; and (2) the number of feasible haplotype configurations a_v of a node v is also bounded by the number of feasible haplotype configurations a_{v_r} of v's neighbor v_r, *i.e.*, $a_v \le 2(\mu_v+1)a_{v_r}$, where μ_v is the number of heterozygous loci in v.

If v is M (or F, respectively), $\alpha_{C_i} = \min\{2^{\mu_{C_i}}, 2(\mu_{C_i}+1)\alpha_v\}$ ($i = 1, \ldots, k$) and $\alpha_F = \min_i\{2^{\mu_F}, 2(\mu_F+1)\alpha_{C_i}\}$. If v is C_i, then $\alpha_F = \min\{2^{\mu_F}, 2(\mu_F+1)\alpha_v\}$, $\alpha_M = \min\{2^{\mu_M}, 2(\mu_M+1)\alpha_v\}$ and $\alpha_{C_i} = \min\{2^{\mu_{C_i}}, 2(\mu_{C_i}+1)\alpha_F, 2(\mu_{C_i}+1)\alpha_M\}$ ($i = 1, \ldots, k$). Thus, the number of feasible haplotype configuration combinations, t_i in trio T_i can be computed consequently, assuming an arbitrary node (node R) as the root of the searching tree. The total number of feasible haplotype configuration combinations in all trios in the pedigree is $t_R = \Sigma_i t_i$, which can be computed in a tree traversal.

Theorem 3. *Let m_0 be the number of heterozygous loci and t_R be the total number of feasible haplotype configuration combinations for all trios in the pedigree with node R as root. Then the node which gives* $\min(t_R)$ *can be found in* $O(n^2 m_0)$ *time and the 1-MRHI problem can be solved in* $O(m_0 \min(t_R))$ *time.*

The diameter of pedigree graphs in many practical instances is usually small. For example, the 452 families in the CEPH database [1][2][3] consist of only three generations. Suppose any node can be reached within l steps from R. We enumerate all 2^{μ_R} feasible haplotype configurations of the root, and no more than $2^{\mu_R} \times 2(m_0+1)$ feasible haplotype configurations for each of its neighboring nodes, and so on, and at most $2^{\mu_R} \times 2^l m_0^{\,l}$ at the most distant node.

Theorem 4. *The 1-MRHI problem can be solved in* $\min(O(nm_0{}^4 2^{m_0}), O(n2^{l+\mu_R} m_0^{\,l+1}))$ *time for a pedigree with n nodes and at most m_0 heterozygous loci in each node, where l is the maximum path length from the root to the leaves and μ_R is the number of heterozygous loci in root R.*

5 Experimental Results

We implemented the above DP algorithm in C++, and all experiments were conducted on a Pentium IV PC with 1.7GHz CPU and 256MB RAM.

5.1 Real Data

We examined real data set Epsiodic Ataxia (EA) by Litt et al.[10] which involves a family containing 29 people typed at 9 polymorphic markers on chromosome 12p. Both the locus-based algorithm [5] and the 1-MRHI algorithm run fast ($t < 1$ *sec.*) on this data set but the results are different. The locus-based algorithm gives a feasible solution with 5 recombinations in total but with a double recombination in one haplotype of member "100". The 1-MRHI algorithm finds a more credible

solution that has 6 recombinations in total, but with at most 1 recombination for each haplotype in the pedigree.

Another two real data sets are three generations families like those in the CEPH database [1][2][3] (*ftp://genome.wi.mit.edu/distribution/mpg/hapmap/ hap_ struct/ popA/* (Gabriel et al.)): family 1331 on chromosome 7a, and family 1346 on chromosome 2a. After removing loci with missing alleles, family 1331 is a pedigree with 8 members on 32 loci, and family 1346 is a pedigree with 8 members on 55 loci. Both the locus-based and the 1-MRHI algorithm give the same answer for family 1331, but take 522.4s and 8.7s, respectively. As for family 1346, the locus-based algorithm fails because of not enough resources while the 1-MRHI algorithm finds a solution in 31 minutes.

5.2 Simulated Data

We compared our algorithm, with the locus-based algorithm [5] and PHASE [14], in terms of *running time t* and *accuracy ratio* ρ (in recovering the correct haplotype configurations for the whole pedigree). We used three different tree pedigree structures in the experiment: (1) a tree with 13 nodes (Figure 1), (2) a tree with 29 nodes (Figure 8 in [8]), and (3) a typical family with 21 nodes from the CEPH database [1][2][3].

Table 1. Comparison of performances of different haplotyping programs on simulation data

(n,r)	$m = 15$						$m = 30$	
	Locus-base		PHASE [13]		1-MRHI		1-MRHI	
	t (sec.)	ρ	*t (sec.)*	ρ*	*t (sec.)*	ρ	*t (sec.)*	ρ
(13, 0.0)	255.7	1.00	688.2	.87	1.68	1.00	202.8	1.00
(29, 0.0)	576.3	1.00	1772.8	.91	12.33	1.00	839.6	1.00
(21, 0.0)	234.4	1.00	592.4	.95	1.02	1.00	44.0	1.00
(13, 0.1)	287.7	.93	972.3	.85	1.73	.91	241.1	.92
(29, 0.1)	542.8	.90	2210.2	.90	10.45	.90	1042.8	.94
(21, 0.1)	243.2	.91	1504.2	.93	0.52	.94	33.7	.96
(13, 0.2)	294.2	.85	1221.4	.85	3.17	.89	1032.4	.86
(29, 0.2)	613.5	.81	3022.2	.89	11.70	.84	916.1	.84
(21, 0.2)	244.1	.90	2106.7	.93	1.22	.95	47.4	.92

[1] *Average performance is obtained from 100 independent executions of each program and for each parameter setting. n stands for the number of nodes, m for the number of marker loci, r for the recombination rate, t(sec.) for the average running time, and* ρ *for the accuracy ratio.*

[2] *The locus-based algorithm and PHASE cannot be applied where* $m \geq 30$ *due to space and time limitations, respectively.*

* *Accuracy ratio of PHASE is defined as the ratio of correctly resolved genotypes to all genotypes.*

For each pedigree, genotypes with 15 and 30 biallelic marker loci are considered. The two alleles at each locus of a founder are independently sampled with a fixed frequency of 0.5. The recombination rate is set to r = 0, 0.1, 0.2 between generations, and the number of recombinations is no more than one in each PO-pair.

As seen from Table 1, 1-MRHI runs quickest and can be applied to larger instances. All three algorithms can recover the correct haplotype configurations with high probability. The accuracy ratio decreases with the increase in the number of recombinations. Since we have limited the number of recombinations within each PO-pair to no more than one, the locus-based algorithm performs worse than the 1-MRHI algorithm as expected.

Acknowledgement

We thank T. Jiang and J. Li for sharing their DP code with us and the RGC Grant supporting this research.

References

[1] The CEPH genotype database. http://www.cephb.fr/.
[2] Dausset J. Cann H. Cohen D, Lathrop M, Lalouel J-M, White R. Centre d'Etude du Polymorphisme Humain (CEPH): collaborative genetic mapping of the human genome. Genomics. 6:575-577, 1990.
[3] Murray, J.C. et al. A comprehensive human linkage map with centimorgan density. Science, 265, 2049-2054 (1994).
[4] A.G. Clark. Inference of haplotypes from {PCR}-amplified samples of diploid populations. Mol. Biol. Evol, 7(2):111-122, 1990.
[5] K. Doi, J. Li, and T.Jiang. Minimum recombinant haplotype configuration on tree pedigree. In Proceedings of the Third International Workshop on Algorithms in Bioinformatics (WABI'03). Lecture Notes in Computer Science no. 2812, pages 339-353, 2003.
[6] Griffiths, W. Gelbart, R. Lewontin, and J. Miller. Modern Genetic Analysis: Integrating Genes and Genomes. W.H. Freeman and Company, N.Y., 2002.
[7] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. J. Computational Biology, 8:305-323, 2001.
[8] J. Li and T. Jiang. Efficient inference of haplotypes from genotypes on a pedigree. J. Bioinfo Comp Biol, 1(1):41-69, 2003.
[9] J. Li and T. Jiang. Efficient rule-based haplotyping algorithms for pedigree data. In Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology (RECOMB'03), pages 197-206, 2003.
[10] M. Litt, P. Kramer, D. Browne, S. Gancher, E.R.P. Brunt, D. Root, et al. A gene for episodic ataxia/myokymia maps to chromosome 12p13. Am J Hum Genet 1994; 55: 702-709.
[11] J.R. O'Connell. Zero-recombinant haplotyping: applications to fine mapping using SNPs Genet Epidemiol, 19:S64-70, 2000.

[12] D. Qian and L. Beckmann. Minimum-recombinant haplotyping in pedigrees. Am J Hum Genet, 70(6): 1434-1445, 2002.
[13] E. Russo et al. Single nucleotide polymorphism: Big pharma hedges its bets. The Scientist, 13(15):1, 1999.
[14] M. Stephens, N.J. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction for population data. Am. J. Hum. Genet, 68:978-989, 2001.
[15] P. Tapadar, S. Ghosh, and P.P. Majumder. Haplotyping in pedigrees via a genetic algorithm. Hum Hered, 50(1):43-56, 2000.

Improved Tag Set Design and Multiplexing Algorithms for Universal Arrays*

Ion I. Măndoiu, Claudia Prăjescu, and Dragoş Trincă

CSE Department, University of Connecticut,
371 Fairfield Rd., Unit 2155, Storrs, CT 06269-2155
{ion.mandoiu, claudia.prajescu, dragos.trinca}@uconn.edu

Abstract. In this paper we address two optimization problems arising in the design of genomic assays based on universal tag arrays. First, we address the universal array tag set design problem. For this problem, we extend previous formulations to incorporate antitag-to-antitag hybridization constraints in addition to constraints on antitag-to-tag hybridization specificity, establish a constructive upper bound on the maximum number of tags satisfying the extended constraints, and propose a simple greedy tag selection algorithm. Second, we give methods for improving the multiplexing rate in large-scale genomic assays by combining primer selection with tag assignment. Experimental results on simulated data show that this integrated optimization leads to reductions of up to 50% in the number of required arrays.

1 Introduction

High throughput genomic technologies have revolutionized biomedical sciences, and progress in this area continues at an accelerated pace in response to the increasingly varied needs of biomedical research. Among emerging technologies, one of the most promising is the use of *universal tag arrays* [4, 7, 9], which provide unprecedented assay customization flexibility while maintaining a high degree of multiplexing and low unit cost.

A universal tag array consists of a set of DNA *tags*, designed such that each tag hybridizes strongly to its own *antitag* (Watson-Crick complement), but to no other antitag [2]. Genomic assays based on universal arrays involve multiple hybridization steps. A typical assay [3, 5], used for Single Nucleotide Polymorphism (SNP) genotyping, works as follows. (1) A set of *reporter oligonucleotide probes* is synthesized by ligating antitags to the $5'$ end of primers complementing the genomic sequence immediately preceding the SNP location in $3'$-$5'$ order on either the forward or reverse strands. (2) Reporter probes are hybridized in solution with the genomic DNA under study. (3) Hybridization of the primer part ($3'$ end) of a reporter probe is detected by a single-base extension reaction using the polymerase enzyme and dideoxynucleotides fluorescently labeled

* Work supported in part by a "Large Grant" from the University of Connecticut's Research Foundation.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 994–1002, 2005.
© Springer-Verlag Berlin Heidelberg 2005

with 4 different dyes. (4) Reporter probes are separated from the template DNA and hybridized to the universal array. (5) Finally, fluorescence levels are used to determine which primers have been extended and learn the identity of the extending dideoxynucleotides.

In this paper we address two optimization problems arising in the design of genomic assays based on the universal tag arrays. First, we address the universal array *tag set design problem* (Section 2). To enable the economies of scale afforded by high-volume production of the arrays, tag sets must be designed to work well for a wide range of assay types and experimental conditions. Ben Dor et al. [2] have previously formalized the problem by imposing constraints on antitag-to-tag hybridization specificity under a hybridization model based on the classical 2-4 rule [10]. We extend the model in [2] to also prevent antitag-to-antitag hybridization and the formation of antitag secondary structures, which can significantly interfere with or disrupt correct assay functionality. Our results on this problem include a constructive upper bound on the maximum number of tags satisfying the extended constraints, as well as a simple greedy tag selection algorithm.

Second, we study methods for improving the multiplexing rate (defined as the average number of reactions assayed per array) in large-scale genomic assays involving multiple universal arrays. In general, it is not possible to assign all tags to primers in an array experiment due to, e.g., unwanted primer-to-tag hybridizations. An assay specific optimization that determines the multiplexing rate (and hence the number of required arrays for a large assay) is the *tag assignment problem*, whereby individual (anti)tags are assigned to each primer. In Section 3 we observe that significant improvements in multiplexing rate can be achieved by combining primer selection with tag assignment. For most universal array applications there are multiple primers with the desired functionality; for example in the SNP genotyping assay described above one can choose the primer from either the forward or reverse strands. Since different primers hybridize to different sets of tags, a higher multiplexing rate is achieved by integrating primer selection with tag assignment. This integrated optimization is shown in Section 4 to lead to a reduction of up to 50% in the number of required arrays.

2 Universal Array Tag Set Design

The main objective of universal array tag set design is to maximize the number of tags, which directly determines the number of reactions that can be multiplexed using a single array. Tags are typically required to have a predetermined length [1, 7]. Furthermore, for correct assay functionality, tags and their antitags must satisfy the following hybridization constraints:

(H1) Every antitag hybridizes strongly to its tag;
(H2) No antitag hybridizes to a tag other than its complement; and
(H3) There is no antitag-to-antitag hybridization (including hybridization between two copies of the same tag and self-hybridization), since the formation of such duplexes and hair-pin structures prevents corresponding reporter

probes from hybridizing to the template DNA and/or leads to undesired primer mis-extensions.

Hybridization affinity between two oligonucleotides is commonly characterized using the *melting temperature*, defined as the temperature at which 50% of the duplexes are in hybridized state. As in previous works [2, 3], we adopt a simple hybridization model to formalize constraints (H1)-(H3). This model is based on the observation that stable hybridization requires the formation of an initial *nucleation complex* between two perfectly complementary substrings of the two oligonucleotides. For such complexes, hybridization affinity is well approximated using the classical *2-4 rule* [10], which estimates the melting temperature of the duplex formed by an oligonucleotide with its complement as the sum between twice the number of A+T bases and four times the number of G+C bases.

The *complement* of a string $x = a_1 a_2 \dots a_k$ over the DNA alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}$ is $\bar{x} = b_1 b_2 \dots b_k$, where b_i is the Watson-Crick complement of a_{k-i+1}. The *weight* $w(x)$ of x is defined as $w(x) = \sum_{i=1}^{k} w(a_i)$, where $w(\mathtt{A}) = w(\mathtt{T}) = 1$ and $w(\mathtt{C}) = w(\mathtt{G}) = 2$.

Definition 1. *For given constants l, h, and c with $l \leq h \leq 2l$, a set of tags $\mathcal{T} \subseteq \{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}^l$ is called* feasible *if the following conditions are satisfied:*

- *(C1) Every tag in $\mathcal{T}$ has weight h or more.*
- *(C2) Every DNA string of weight c or more appears as substring at most once in the tags of $\mathcal{T}$.*
- *(C3) If a DNA string x of weight c or more appears as a substring of a tag, then $\bar{x}$ does not appear as a substring of a tag unless $x = \bar{x}$.*

The constants l, h, and c depend on factors such as array manufacturing technology and intended hybridization conditions. Property (H1) is implied by (C1) when h is large enough. Similarly, properties (H2) and (H3) are implied by (C1) and (C2) when c is small enough: constraint (C2) ensures that nucleation complexes do not form between antitags and non-complementary tags, while constraint (C3) ensures that nucleation complexes do not form between pairs of antitags.

Universal Array Tag Set Design Problem: *Given constants l, h, and c with $l \leq h \leq 2l$, find a feasible tag set of maximum cardinality.*

Ben-Dor et al. [2] have recently studied a simpler formulation of the problem in which tags of unequal length are allowed and only constraints (C1) and (C2) are enforced. For this simpler formulation, Ben-Dor et al. established a constructive upperbound on the optimal number of tags, and gave a nearly optimal tag selection algorithm based on De Bruijn sequences. Here, we refine the techniques in [2] to establish a constructive upperbound on the number of tags of a feasible set for the extended problem formulation, and propose a simple greedy algorithm for constructing feasible tag sets.

The constructive upperbound is based on counting the minimal strings, called *c-tokens*, that can occur as substrings only once in the tags and antitags of a feasible set. Formally, a DNA string x is called c-token if the weight of x is c or more, and every proper suffix of x has weight strictly less than c. The *tail weight* of a c-token is defined as the weight of its last letter. Note that the weight of a

c-token can be either c or $c+1$, the latter case being possible only if the c-token starts with a G or a C. As in [2], we use G_n to denote the number of DNA strings of weight n. It is easy to see that $G_1 = 2$, $G_2 = 6$, and $G_n = 2G_{n-1} + 2G_{n-2}$; for convenience, we also define $G_0 = 1$.

Theorem 1. *For every l, h, c with $l \leq h \leq 2l$ and $c \geq 4$, the number of tags in a feasible tag set is at most*

$$\min\left\{\frac{3G_{c-2} + 6G_{c-3} + G_{\frac{c-3}{2}}}{l-c+1}, \frac{2G_{c-1} + 4G_{c-3} + 2G_{\frac{c-3}{2}}}{h-c+1}\right\}$$

for c odd, and at most

$$\min\left\{\frac{3G_{c-2} + 6G_{c-3} + \frac{1}{2}G_{\frac{c}{2}}}{l-c+1}, \frac{2G_{c-1} + 4G_{c-3} + G_{\frac{c-2}{2}} + 2G_{\frac{c-4}{2}}}{h-c+1}\right\}$$

for c even.

Proof. Omitted due to space constraints, see [8] for details.

We employ a simple greedy algorithm to generate feasible sets of tags; a similar algorithm is suggested in [7] for finding sets of tags that satisfy an unweighted version of constraint (C2). We start with an empty set of tags and an empty tag prefix. In every step we try to extend the current tag prefix t by an additional A. If the added letter completes a c-token or a complement of a c-token that has been used in already selected tags or in t itself, we try the next letter in the DNA alphabet, or backtrack to a previous position in the prefix when no more letter choices are left. Whenever we succeed generating a complete tag, we save it and backtrack to the last letter of its first c-token.

3 Improved Multiplexing by Integrated Primer Selection and Tag Assignment

Although constraints (H2)-(H3) in Section 2 prevent unintended antitag-to-tag and antitag-to-antitag hybridizations, the formation of nucleation complexes involving (portions of) the primers may still lead to undesired hybridization between reporter probes and tags on the array (Figure 1(a)), or between two reporter probes (Figure 1(b)-(d)). The formation of these duplexes must be avoided

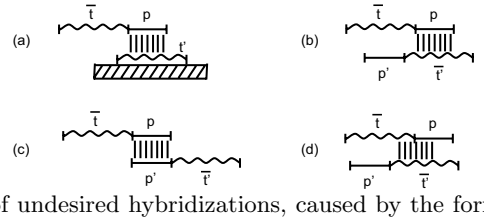


Fig. 1. Four types of undesired hybridizations, caused by the formation of nucleation complexes between (a) a primer and a tag other than the complement of the ligated antitag, (b) a primer and an antitag, (c) two primers, and (d) two reporter probe substrings, at least one of which straddles a ligation point

as it leads to extension misreporting, false primer extensions, and/or reduced effective reporter probe concentration available for hybridization to the template DNA or to the tags on the array [3]. This can be done by leaving some of the tags unassigned. As in [3], we focus on preventing primer-to-tag hybridizations (Figure 1(a)). Our algorithms can be easily extended to prevent primer-to-antitag hybridizations (Figure 1(b)); a simple practical solution for preventing the other (less-frequent) unwanted hybridizations is to re-assign offending primers in a post-processing step.

Following [3], a set $\mathcal{P}$ of primers is called *assignable* to a set $\mathcal{T}$ of tags if there is a one-to-one mapping $a : \mathcal{P} \to \mathcal{T}$ such that, for every tag t hybridizing to a primer $p \in \mathcal{P}$, either $t \not\in \mathcal{A}(\mathcal{P})$ or $t = a(p)$.

Universal Array Multiplexing Problem: *Given primers $\mathcal{P} = \{p_1, \ldots, p_m\}$ and tag set $\mathcal{T} = \{t_1, \ldots, t_n\}$, find a partition of $\mathcal{P}$ into the minimum number of assignable sets.*

For most universal array applications there are multiple primers with the desired functionality, e.g., for the SNP genotyping assay described in Section 1, one can choose the primer from either the forward or reverse strands. Since different primers have different hybridization patterns, a higher multiplexing rate can in general be achieved by integrating primer selection with tag assignment. A similar integration has been recently proposed in [6] between probe selection and physical DNA array design, with the objective of minimizing unintended illumination in photo-lithographic manufacturing of DNA arrays. The idea in [6] is to modify probe selection tools to return *pools* containing all feasible candidates, and let subsequent optimization steps select the candidate to be used from each pool. In this paper we use a similar approach. We say that a set of primer pools is *assignable* if we can select a primer from each pool to form an assignable set of primers.

Pooled Universal Array Multiplexing Problem: *Given primer pools $\mathcal{P} = \{P_1, \ldots, P_m\}$ and tag set $\mathcal{T} = \{t_1, \ldots, t_n\}$, find a partition of $\mathcal{P}$ into the minimum number of assignable sets.*

Let $\mathcal{P}$ be a set of primer pools and $\mathcal{T}$ a tag set. For a primer p (tag t), $\mathcal{T}(p)$ (resp. $\mathcal{P}(t)$) denotes the set of tags (resp. primers of $\bigcup_{P \in \mathcal{P}} P$) hybridizing with p (resp. t). Let $X(\mathcal{P}) = \{P \in \mathcal{P} \ : \ \exists p \in P,\ t \in \mathcal{T}$ s.t. $t \in \mathcal{T}(p)$ and $\mathcal{P}(t) \subseteq P\}$ and $Y(\mathcal{P}) = \{t \in \mathcal{T} \ : \ \mathcal{P}(t) = \emptyset\}$. Clearly, in every pool of $X(\mathcal{P})$ we can find a primer p that hybridizes to a tag t which is not cross-hybridizing to primers in other pools, and therefore assigning t to p will not violate (A1). Furthermore, any primer can be assigned to a tag in $Y(\mathcal{P})$ without violating (A1). Thus, a set $\mathcal{P}$ with $|X(\mathcal{P})| + |Y(\mathcal{P})| \geq |\mathcal{P}|$ is always assignable. The converse is not necessarily true (see [8] for a simple counter-example).

Our primer pool assignment algorithm (see Figure 2) is a generalization to primer pools of Algorithm B in [3]. In each iteration, the algorithm checks whether $|X(\mathcal{P}')| + |Y(\mathcal{P}')| \geq |\mathcal{P}'|$ for the remaining set of pools $\mathcal{P}'$. If not, a primer of maximum *potential* is deleted from the pools. As in [3], the potential of a tag t with respect to $\mathcal{P}'$ is $2^{-|\mathcal{P}'(t)|}$, and the potential of a primer p is the sum of potentials for the tags in $\mathcal{T}(p)$. If the algorithm deletes the last primer

Input: Primer pools $\mathcal{P} = \{P_1, \ldots, P_m\}$ and tag set $\mathcal{T}$
Output: Triples (p_i, t_i, k_i), $1 \leq i \leq m$, where $p_i \in P_i$ is the selected primer for pool i, t_i is the tag assigned to p_i, and k_i is the index of the array on which p_i is assayed

---

$k \leftarrow 0$
While $\mathcal{P} \neq \emptyset$ **do**
  $k \leftarrow k + 1$; $\mathcal{P}' \leftarrow \mathcal{P}$
  While $|X(\mathcal{P}')| + |Y(\mathcal{P}')| < |\mathcal{P}'|$ **do**
    Remove the primer p of maximum potential from the pools in $\mathcal{P}'$
    If p's pool becomes empty **then** remove it from $\mathcal{P}'$
  End While
  Assign pools in $\mathcal{P}'$ to tags on array k
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{P}'$
End While

Fig. 2. The iterative primer deletion algorithm

in a pool P, then P is itself deleted from $\mathcal{P}'$; deleted pools are subsequently assigned to new arrays using the same algorithm.

4 Experimental Results

Tag Set Selection. The greedy tag set design algorithm described in Section 2 can be used to fully or selectively enforce the constraints in Definition 1. In order to assess the effect of various hybridization constraints on tag set size, we ran the algorithm both with constraints (C1)+(C2) and and with constraints (C1)+(C2)+(C3). For each set of constraints, we ran the algorithm with c between 8 and 10 for typical practical requirements [1, 7] that all tags have length 20 and weight between 28 and 32 (corresponding to a GC-content between 40-60%). We also ran the algorithm with the tag length and weight requirements enforced individually.

Table 1 gives the size of the tag set found by the greedy algorithm, as well as the number of c-tokens appearing in selected tags. We also include the theoretical upper-bounds on these two quantities; the upper-bounds for (C1)+(C2) follow from results of [2], while the upper-bounds for (C1)+(C2)+(C3) follow from Theorem 1. The results show that, for any combination of length and weight requirements, imposing the antitag-to-antitag hybridization constraints (C3) roughly halves the number of tags selected by the greedy algorithm – as well as the theoretical upperbound – compared to only imposing antitag-to-tag hybridization constraints (C1)+(C2). For a fixed set of hybridization constraints, the largest tag sets are found by the greedy algorithm when only the length requirement is imposed. The tag weight requirement, which guarantees similar melting temperatures for the tags, results in a 10-20% reduction in the number of tags. However, requiring that the tags have *both* equal length and similar weight results in close to halving the number of tags. This strongly suggests reassessing the need for the strict simultaneous enforcement of the two constraints in current industry designs [1]; our results indicate that allowing small variations in tag length and/or weight results in significant increases in the number of tags.

Table 1. Tag Sets Selected by the Greedy Algorithm

l	$h_{min}/$ h_{max}	c	(C1)+(C2)				(C1)+(C2)+(C3)			
			tags	Bound	c-tokens	Bound	tags	Bound	c-tokens	Bound
20	–/–	8	213	275	2976	3584	107	132	1480	1726
		9	600	816	7931	9792	300	389	3939	4672
		10	1667	2432	20771	26752	844	1161	10411	12780
–	28/32	8	175	224	2918	3584	90	109	1489	1726
		9	531	644	8431	9792	263	312	4158	4672
		10	1428	1854	21707	26752	714	896	10837	12780
20	28/32	8	108	224	1548	3584	51	109	703	1726
		9	333	644	4566	9792	164	312	2185	4672
		10	851	1854	11141	26752	447	896	5698	12780

Integrated Primer Selection and Tag Assignment. We have implemented the iterative primer deletion algorithm in Figure 2 (Primer-Del), a variant of it in which primers in pools of size 1 are omitted – unless all pools have size 1 – when selecting the primer with maximum potential for deletion (Primer-Del+), and two simple heuristics that first select from each pool the primer of minimum potential (Min-Pot), respectively minimum degree (Min-Deg), and then run the iterative primer deletion algorithm on the resulting pools of size 1. We ran all algorithms on data sets with between 1000 to 5000 pools of up to 5 randomly generated primers. As in [3], we varied the number of tags between 500 and 2000.

For instance size, we report the number of arrays and the average tag utilization (computed over all arrays except the last) obtained by (a) algorithm B in [3] run using a single primer per pool, (b) the four pool-aware assignment algorithms run with 1 additional candidate in each pool, and (c) the four pool-aware assignment algorithms run with 4 additional candidates in each pool. Scenario (b) models SNP genotyping applications in which the primer can be selected from both strands of the template DNA, while scenario (c) models applications such as gene transcription monitoring, where significantly more than 2 gene specific primers are typically available.

We performed experiments with tag sets both constructed by running the greedy algorithm in Section 2 and extracted from the commercially available GenFlex Tag Arrays. Due to space constraints we report here only results for GenFlex tags, for additional results see [8]. All GenFlex tags have length 20; primers used in our experiments are 20 bases long as well. Primer-to-tag hybridizations were assumed to occur between primers and tags containing complementary c-tokens with $c = 7$. The results in Table 2 show that significant improvements in multiplexing rate – and a corresponding reduction in the number of arrays – are achieved by the pool-aware algorithms over the algorithm in [3]. For example, assaying 5000 reactions on a 2000-tag array requires 18 arrays using the method in [3] for $c = 7$, compared to only 13 (respectively 9) if 2 (respectively 5) primers per pool are available. In these experiments, the Primer-Del+ algorithm dominates in solution quality the Primer-Del, while Min-Deg dominates Min-Pot. Neither Primer-Del+ nor Min-Deg consistently

Table 2. Multiplexing results for $c = 7$ (averages over 10 test cases)

# pools	Pool size	Algorithm	500 tags #arrays	500 tags % Util.	1000 tags #arrays	1000 tags % Util.	2000 tags #arrays	2000 tags % Util.
1000	1	[3]	7.5	30.1	6.0	19.3	5.0	12.1
	2	Primer-Del	6.0	38.7	5.0	24.3	4.1	15.5
	2	Primer-Del+	6.0	39.6	4.5	27.3	4.0	16.5
	2	Min-Pot	6.0	38.4	5.0	24.2	4.0	15.9
	2	Min-Deg	5.8	40.9	4.6	27.0	4.0	16.4
	5	Primer-Del	5.0	49.6	4.0	32.5	3.3	21.0
	5	Primer-Del+	4.0	60.4	3.0	43.6	3.0	24.7
	5	Min-Pot	4.9	50.6	4.0	33.0	3.0	23.5
	5	Min-Deg	4.0	62.0	3.0	44.9	2.7	28.1
5000	1	[3]	29.5	35.0	23.0	22.6	18.0	14.6
	2	Primer-Del	22.2	47.0	17.1	30.9	13.7	19.6
	2	Primer-Del+	22.2	46.8	17.0	30.9	13.1	20.4
	2	Min-Pot	25.0	41.5	19.2	27.3	15.0	17.7
	2	Min-Deg	22.0	47.3	17.0	31.0	13.0	20.6
	5	Primer-Del	16.6	63.8	12.3	43.9	10.0	27.8
	5	Primer-Del+	16.0	65.6	12.0	44.9	9.0	30.6
	5	Min-Pot	29.5	35.0	23.0	22.6	18.0	14.6
	5	Min-Deg	16.0	65.8	12.0	45.2	9.0	30.8

outperforms the other over the whole range of parameters, which suggests that a good practical meta-heuristic is to run both of them and pick the best solution obtained.

References

1. Affymetrix, Inc. GeneFlex tag array technical note no. 1, available online at http://www.affymetrix.com/support/technical/technotes/genflex_technote.pdf.
2. A. Ben-Dor, R. Karp, B. Schwikowski, and Z. Yakhini. Universal DNA tag systems: a combinatorial design scheme. *Journal of Computational Biology*, 7(3-4):503–519, 2000.
3. A. BenDor, T. Hartman, B. Schwikowski, R. Sharan, and Z. Yakhini. Towards optimally multiplexed applications of universal DNA tag systems. In *Proc. 7th Annual International Conference on Research in Computational Molecular Biology*, pages 48–56, 2003.
4. S. Brenner. Methods for sorting polynucleotides using oligonucleotide tags. *US Patent 5,604,097*, 1997.
5. J.N. Hirschhorn et al. SBE-TAGS: An array-based method for efficient single-nucleotide polymorphism genotyping. *PNAS*, 97(22):12164–12169, 2000.
6. A.B. Kahng, I.I. Măndoiu, S. Reda, X. Xu, and A. Zelikovsky. Design flow enhancements for DNA arrays. In *Proc. IEEE International Conference on Computer Design (ICCD)*, pages 116–123, 2003.
7. M.S. Morris, D.D. Shoemaker, R.W. Davis, and M.P. Mittmann. Selecting tag nucleic acids. *U.S. Patent 6,458,530 B1*, 2002.

8. I.I. Măndoiu, C. Prăjescu, and D. Trincă. Improved tag set design and multiplexing algorithms for universal arrays. ACM Computing Research Repository, cs.DS/0502054, 2005.
9. N.P. Gerry et al. Universal DNA microarray method for multiplex detection of low abundance point mutations. *J. Mol. Biol.*, 292(2):251–262, 1999.
10. R.B. Wallace, J. Shaffer, R.F. Murphy, J. Bonner, T. Hirose, and K. Itakura. Hybridization of synthetic oligodeoxyribonucleotides to phi chi 174 DNA: the effect of single base pair mismatch. *Nucleic Acids Res.*, 6(11):6353–6357, 1979.

A Parallel Implementation for Determining Genomic Distances Under Deletion and Insertion

Vijaya Smitha Kolli[1], Hui Liu[1], Michelle Hong Pan[2], and Yi Pan[1]

[1] Department of Computer Science, Georgia State University,
Atlanta, GA 30303, USA
hui.anitaliu@gmail.com, pan@cs.gsu.edu
[2] Centers for Disease Control and Prevention, Office of Workforce
and Career Development, Career Development Division,
Public Health Informatics Fellow Program, 1600 Clifton Rd,
Atlanta, GA 30333, USA
hdp1@cdc.gov

Abstract. As the need for comparing genomes of different species has grown dramatically with the fast progress of the Human Genome Project, the evolution at the level of whole genomes has attracted more and more attention from both biologists and computer scientists. They are especially interested in the scenarios in which the genome evolves through insertions, deletions, and movements of genes along its chromosomes. Marron et al proposed a polynomial-time approximation algorithm to compute (near) minimum edit distances under inversions, deletions, and unrestricted insertions. Our work is based on their algorithm, which carries out lots of comparisons and sorting to calculate the edit distance. These comparisons and sorting are extremely time-consuming, and they result in decrease of computational efficiency. We believe the time of the algorithm can be improved through parallelization. We parallelize their algorithm via OpenMP using Intel C++ compiler for Linux 7.1, and compare three levels of parallelism: coarse grain, fine grain and combination of both. The experiments are conducted for a varying number of threads and different lengths of the gene sequences. The experimental results show that either coarse grain parallelism or fine grain parallelism alone does not improve the performance of the algorithm very much. However, the use of combination of both fine grain and coarse grain parallelism improves the performance of the algorithm drastically.

1 Introduction

A gene is the fundamental physical and functional unit of heredity. Each chromosome can be represented by an ordering of signed genes. The gene orders can be rearranged via evolutionary events like inversions and transpositions. The motivation of studying the gene sequencing arises in molecular biology. The ability to compare genomes of different species has grown considerably with the rapid advancement of Human Genome Project. One of the most effective methods of finding the similarity between genomes is to compare the order of appearance of identical genes in the two species.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1003 – 1010, 2005.
© Springer-Verlag Berlin Heidelberg 2005

By finding the corresponding shortest edit distance between two signed gene permutations, we can know the smallest number of insertions, deletions, and inversions required to change one string of genes into another, where insertion, deletion and inversion are the process of adding, deleting and reversing a single or sequence of genes in the existing gene sequence respectively. However, it is very difficult to compute the edit distance between two genomes; for example, this problem is NP-hard for unsigned permutations even with equal gene content and only inversion allowed.

Many researchers have proposed various algorithms on finding the minimum edit distances [3,4,5,6,10]. Most of these algorithms involve lots of comparisons and sorting while computing edit distances. Marron et al's algorithm [2] is of particular interest for our implementation purposes. This algorithm handles duplications as well as insertions and presents an alternate framework for computing (near) minimal edit sequences involving insertions, deletions, and inversions.

Marron et al's algorithm is implemented sequentially, which is not efficient in terms of computation time, because the algorithm carries out many comparisons and sorting. Parallelism can be employed in the time-consuming comparisons and sorting, thus increasing the efficiency of the algorithm. We have performed profiling on the whole algorithm and identified the functions that are utilizing maximum time. We use OpenMP on Intel® C++ for Linux compiler 7.1 to parallelize the algorithm [9].

The rest of this paper proceeds as follows. Section 2 provides preliminaries on the sequential algorithm by Marron et al. We illustrate the details of our parallel implementation in Section 3. The experimental results are analyzed in Section 4. Section 5 draws the conclusion and proposes future works.

2 Preliminary Existing Algorithm

Marron et al's algorithm [2] is based on a new canonical form for edit sequences. They showed that shortest edit sequences can be transformed into equivalent sequences of equal length in which all insertions are performed first, followed by all inversions, and then by all deletions. This canonical form allows taking advantage of El-Mabrouk's exact algorithm for inversions and deletions, which can be extended by finding the best possible prefix of insertions, producing an approximate solution with bounded error.

2.1 Canonical Form

Marron et al [2] have proved some positive results about shortest edit sequences. These results will enable to obtain a "*canonical form*" into which any shortest edit sequence can always be transformed without losing optimality. Marron et al has proved the following two theorems.

Theorem 1: One two substrings become correctly oriented, they remain correctly oriented.

Theorem 2: All insertions can be done before all inversions and deletions in a Minimum Edit Sequence.

2.2 Sequences Cover

A group of substrings from the target should be determined such that every element in the source appears in one of those substrings. The goal of the job is to cover all the non-deleted target elements with one from the subject. A minimal cover is one that uses fewest substrings of the subject. At each step, we try to cover the target from the left to as far as right as possible with contiguous subsequences of the subject. At last, this method produces a minimal cover by a greedy algorithm. The cover bound is proved in theorem 3.

Theorem 3: There exists a cover of at most $2|\pi|+1$ for a sequence of S.

Now we can summarize Marron et al's Genomic Distances algorithm [2] in the following three steps.

Step 1: Relocate insertions to obtain the canonical form of sequences;
Step 2: Resolve duplicates by finding the minimum cover through greedy method;
Step 3: Then run exact EI-Mabrouk algorithm on the inversions and deletions.

3 Details of Parallel Implementation

This section gives the details of the implementation steps carried out to parallelize the Marron et al's genomic distances algorithm.

3.1 Profiling

Profiling is a good procedure to determine the most time-consuming parts of the code. By profiling the sequential code, we obtain a flat profile. Flat profile consists of the percentage of time used to complete the particular function, number of calls made for the function, self milli seconds per call and total milli seconds per call. From this flat profile, we can deduce which function uses greater percentage of time and further which function consumes more total milli seconds per call. The *gprof* utility provides a fast and easy way to do procedural-level profiling of the code. We use *gprof* and focus on the identification of such expensive functions with respect to time, and acknowledge the code to be parallelized. So that, these functions may utilize less time and eventually good *speedup* can be obtained. We compile the code using `-pg` option and run the code as usual. Then the code will produce an output file named gmon.out, which can be analyzed for further purposes.

3.2 Porting the Code Between GNU 3.2 and Intel C++ Compiler 7.1 for Linux

In order to parallelize the algorithm using OpenMP we need to compile the program on Intel C++ compiler because GNU 3.2 does not support OpenMP. There are some significant porting issues encountered while porting program from GNU 3.2 to Intel C++ compiler 7.1. The header files in GNU 3.2 compiler are declared with extension .h as #include < stack.h > but in Intel C++ compiler it is declared without .h extension as #include <stack>.

The initialization of the hash_map variables is much different in GNU compiler and Intel C++ compiler. In GNU, the hash_map is initialized with three parameters, while in Intel C++ compiler it does not accept three parameters in the initialization. Due to this, the code has to be changed to compile successfully on the Intel C++ Compiler without the third parameter.

3.3 Parallel Implementation of Genomic Distances Algorithm

Genomic Distances algorithm is parallelized by using OpenMP on Intel C++ compiler 7.1 for Linux. Parallelism is implemented by applying fine-grain parallelism, coarse-grain parallelism and combination of both. OpenMP method makes use of fork-join technique. Master thread spawns team of threads as needed.

In coarse-grain parallelism, each function is given attention to find the functional dependencies, which is needed for synchronization, and mutual exclusive sections. Then the independent functions are taken and parallelized using the *parallel* and *section* directive. The mutually exclusive sections are provided with *critical section* directive, which specifies a region of code that must be executed by only one thread at a time.

In order to implement fine grain parallelism, first data dependency and critical sections within loops is checked to determine whether variables in nested loops should be declared as private or shared in OpenMP program after profiling and porting the program. The variables in the for loops are limited to individual threads, therefore they should be declared as private using *private* directive. Every thread has its own copy of these variables in parallel loops. A variable without the declaration is treated as shared variables by default. Second, the appropriate directive *parallel for* is added in front of loops to inform compilers to execute the code in parallel mode. All threads are scheduled dynamically with varying chunk sizes.

To implement the combination of coarse-grain and fine-grain parallelism in the Genomic Distances algorithm, both the *parallel section* and *parallel for* directive are applied appropriately in the required regions. Special attention is paid to mutual exclusive section.

4 Analysis of Experimental Results

This section discusses the experimental environment and results obtained in implementation of Genomic Distances algorithm via OpenMP. The Algorithm is

parallelized by using fine-grain, coarse-grain and combination of both coarse and fine grain parallelism.

4.1 Experimental Environment

We test OpenMP performance by Intel C++ compiler 7.1. This compiler has advanced optimization techniques for the Intel processor. The machine used is the Dell poweredge 6600 server and it has four quad SMP CPU's. Dell server is a powerful scalable parallel system. Experiments are conducted by varying the number of threads (2, 4, 6, 8) and length of gene sequences (400, 800, 1000, 2500, 5000, 10000). Speedup and program execution time for OpenMP are measured. Experiment results are verified and made sure that the identical gene sequences are utilized for sequential and parallel program.

4.2 Coarse-Grain Parallelism

As a first step to improve the efficiency of the Genomic Distance algorithm, we adopt OpenMP *parallel sections* construct for all the time-consuming functions. Special attention is paid to the functional dependencies. Fig. 1 compares sequential timing with the coarse-grain parallelism timing with varying the number of threads and length of gene sequences.

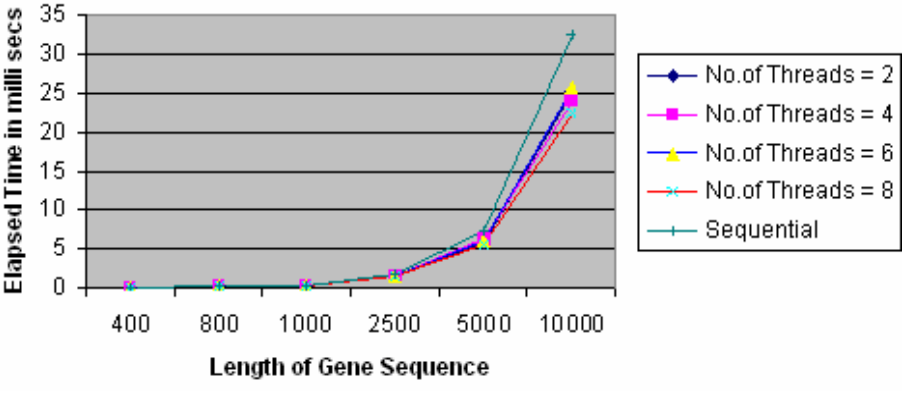


Fig. 1. Average execution time relative to the length of gene sequences under coarse-grain parallelism

Here the sequential algorithm was the most expensive in terms of execution time. We will compare the speedup values among three cases; coarse-grain parallelism, fine-grain parallelism, and the combination of coarse-grain and fine-grain parallelism when the length of gene sequence is 10000, and the number of threads is 8. For coarse-grain parallelism, the speedup value is 1.32. Thus, we conclude that by parallelizing the algorithm with the *parallel section* construct there is not much

improvement in execution time. Since there is a lot of dependency in the code, coarse-grain parallelism by sections does not yield a great performance progress.

4.3 Fine-Grain Parallelism

Later, to improve the efficiency of the Genomic Distance algorithm, we use OpenMP *parallel for* construct for all the time-consuming loops in the program. As the *for* directive specifies that the iterations of the loop immediately following it must be executed in parallel by the team of threads, each for loop will divide the array used into chunks of the specified size dynamically, and the number of threads will work on each individual chunk. With varying sequence size, care is taken to make the chunk size equally distributable among the available threads. Fig. 2 shows the results obtained as a part of fine-grain parallelism.

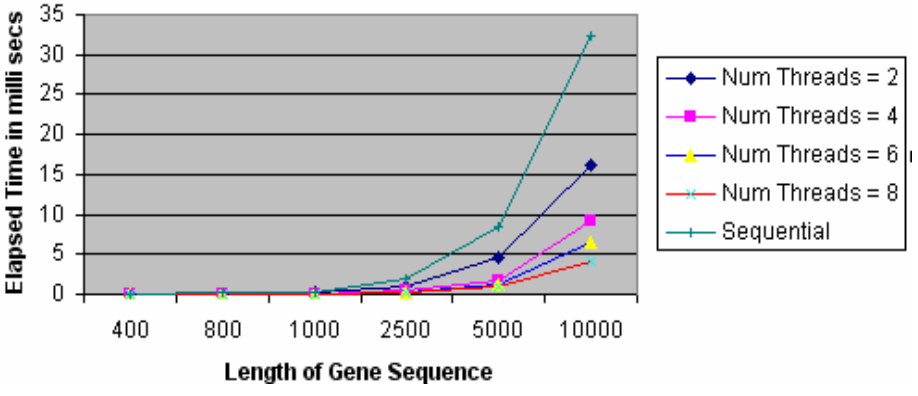


Fig. 2. Average execution time relative to the length of gene sequences under fine-grain parallelism

Here the sequential algorithm is the most consuming in terms of execution time. However, by parallelizing the algorithm with the *parallel for* construct there is a certain improvement. For example, the speedup value is 5.91 when gene sequence length is 10000, and the number of treads is 8. This is a good improvement compared with only coarse-grain parallelism.

4.4 The Combination of Coarse-Grain Parallelism and Fine-Grain Parallelism

Finally, to get a more efficient Genomic Distance algorithm, we adopt both OpenMP fine-grain and coarse-grain parallelism by implementing both *parallel for* construct and *parallel sections* construct to all the time consuming loops and functions. Fig. 3 shows the results obtained as a part of both fine-grain and coarse-grain parallelism.

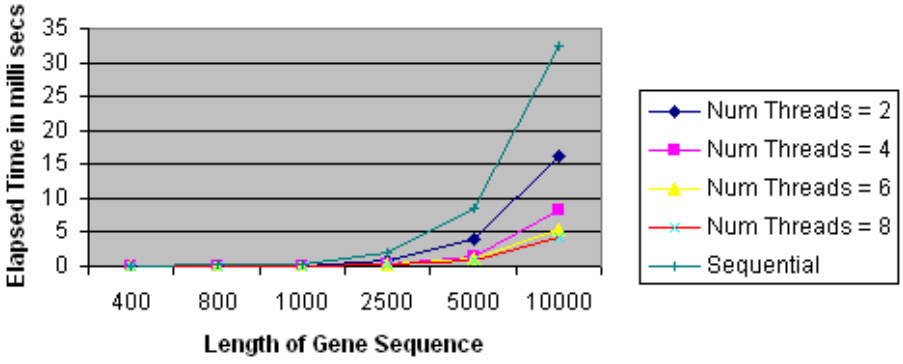


Fig. 3. Average execution time relative to the length of gene sequences under the combination of coarse-grain parallelism and fine-grain parallelism

As in the earlier cases, the figure shows that the sequential algorithm is the most consuming in terms of execution time. On the other hand, by parallelizing the algorithm with the *parallel for* and *parallel sections* constructs there is a very good improvement. For example, the speedup is 7.36 with gene sequence length 10000 and eight threads, which has achieved much better performance compared with only coarse-grain parallelism and only fine-grain parallelism.

By observing all the above three cases, we notice that the combination of both fine-grain and coarse-grain parallelism has improved the performance of the algorithm to a great extent, because it implements both loop level parallelism, which takes care about all the time-consuming loops, and functional level parallelism, which handles the independent functions and execute them in parallel.

5 Conclusion and Future Work

The Genomic Distances algorithm in [2] provides a polynomial-time approximation algorithm with bounded error to compute edit distances under inversions, deletions, and unrestricted insertions from the perfectly sorted sequence to any other. The algorithm consists of many comparisons and sorting, so it is extremely time-consuming. In order to improve the efficiency of the algorithm, we parallelize the algorithm using OpenMP. Furthermore, we study extensively the performance metrics for fine-grain and coarse-grain parallelism and both together. Since we can solve the problem quickly, we can handle larger gene sequences which are very important in practices in bioinformatics research.

From our experimental results, we conclude that coarse grain parallelism is not effective for this algorithm since there is lot of functional dependencies, and many functions are not able to execute concurrently. There is improvement in performance when the algorithm is parallelized with fine grain parallelism, for all the time-consuming for loops are made to run in parallel. When the algorithm is parallelized by

the combination of both fine and coarse grain parallelism, there is very good improvement in the efficiency of the algorithm.

In the future, we can obtain better performance of the Genomic Distances algorithm by using both MPI and OpenMP. MPI handles the larger-grained communications among multiprocessors, while the lighter-weight threads of OpenMP handle the processor interactions within each multiprocessor. By adding MPI function calls to the OpenMP source program, the program can be transformed into a MPI/OpenMP program suitable for execution on a cluster of multiprocessors.

References

1. D.A. Bader, B.M.E. Moret, and M. Yan. A fast linear-time algorithm for inversion distance with an experimental comparison. *J. Comput. Biol.*, 8(5):483. 491, 2001.
2. Mark Marron, Krister M. Sweson, and Bernard M. E. Moret. Genomic Distances under Deletions and Insertions. *Proc. 9th Int'l Combinatorics and Computing Conf. COCOON'03*, 2003, 537-547.
3. A. Caprara. Sorting by reversals is difficult. In *Proc. 1st Int'l Conf. on Comput.Mol. Biol. RECOMB97*, pages 75.83. ACM Press, 1997.
4. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. 27th Ann.Symp. Theory of Computing STOC 95*, pages 178.189. ACM Press, 1995.
5. N. El-Mabrouk. Genome rearrangement by reversals and insertions/deletions of contiguous segments. In *Proc. 11th Ann. Symp. Combin. Pattern MatchingCPM 00*, volume 1848 of *Lecture Notes in Computer Science*, pages 222.234. Springer-Verlag, 2000.
6. T. Liu, B.M.E. Moret, and D.A. Bader. An exact linear-time algorithm for computing genomic distances under inversions and deletions *U. New Mexico*, TR-CS-2003-31.
7. Tutorial on OpenMP, http://www.llnl.gov/computing/tutorials/openMP/
8. Michael Quinn, *Parallel Programming in C with MPI and OpenMP*, The McGraw-Hill Companies, 2004.
9. Intel C++ Compiler for Linux, http://www.intel.com/software/products/compilers/clin/clinux.htm
10. Haim Kaplan, Ron Shamir and Robert E. Tarjan, "Faster and Simpler Algorithm for Sorting Signed Permutations by Reversals", *Proc. SODA 97 pages 344—351,SIAM Journal on Computing* 29 (3) 880--892 (1999).

Phasing and Missing Data Recovery in Family Trios*

Dumitru Brinza, Jingwu He**, Weidong Mao, and Alexander Zelikovsky***

Department of Computer Science, Georgia State University, Atlanta, GA 30303
{dima, jingwu, weidong, alexz}@cs.gsu.edu

Abstract. Although there exist many phasing methods for unrelated adults or pedigrees, phasing and missing data recovery for data representing family trios is lagging behind. This paper is an attempt to fill this gap by considering the following problem. Given a set of genotypes partitioned into family trios, find for each trio a quartet of parent haplotypes which agree with all three genotypes and recover the SNP values missed in given genotype data. Our contributions include (i) formulating the pure-parsimony trio phasing and the trio missing data recovery problems, (ii) proposing two new greedy and integer linear programming based solution methods, and (iii)extensive experimental validation of proposed methods showing advantage over the previously known methods.

Keywords: haplotypes, genotypes, SNP, family trio data, phasing.

1 Introduction

In diploid organisms (such as human) each chromosome has two "copies" which are not completely identical. Each of two single copies is called a haplotype, while a description of the data consisting of mixture of the two haplotypes is called a genotype. The underlying data that forms a haplotype is either the full DNA sequence in the region, or more commonly the values of single nucleotide polymorphisms (SNPs) in that region. A SNP is a single nucleotide site where two or more (out of four) different nucleotides occur in a large percentage of the population.

In general, it is costly and time consuming to examine the two copies of a chromosome separately, and genotype data rather than haplotype data are only available, even though it is the haplotype data that will be of greatest use for investigating complex diseases. Data from m sites (SNPs) in n individual genotype are collected, where each site has one of two alleles denoted by 0 and 1 (the case of three- and four-allelic SNPs can be reduced to the biallelic case). The input to the phasing problem consists of n genotype vectors, each of length m,

* Partially supported by NIH Award 1 P20 GM065762-01A1.
** Supported by GSU Molecular Basis of Disease Fellowship.
*** Authors are listed in alphabetical order.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1011–1019, 2005.
© Springer-Verlag Berlin Heidelberg 2005

where each value in the vector is either 0,1, or 2. The position in the genotype vector has a value of 0 or 1 if the associated chromosome site has that allele on both copies (it is a homozygous site), and has a value of 2 otherwise (the chromosome site is heterozygous).

Usually, genotype data miss certain SNP values due to failures during genotyping. Although the missing data rate decreases, still unavailable SNP values constitute a substantial part of the entire data (as large as 16% of the genotype data in Daly et al. [4] data and 10% in Gabriel et al. [5]). The problem of missing data recovery attracts a great deal of attention [9, 14].

Commonly, genotype data represent family trios consisting of the two parents and their offspring since that allows to recover haplotypes with higher confidence. A simple logical analysis allows to substantially decrease uncertainty of phasing. For example, for two SNPs in a trio with parent genotypes $f = 22$ and $m = 02$, and the offspring genotype $k = 01$, there is a unique feasible phasing of the parents: $f_1 = 10$, $f_2 = 01$, $m_1 = 01$, $m_2 = 00$ such that the haplotypes f_2 and m_1 are inherited by the offspring. In fact, it is not difficult to check that logical ambiguity exists only if all three genotypes have 2's in the same SNP site.

Although there exist many phasing methods for unrelated adults or pedigrees, phasing and missing data recovery for trios is lagging behind. We have tried several well-known computational methods for phasing Daly et al. [4] family trio data, but, surprisingly, all of them give infeasible solutions with high inconsistency rate.

This paper deals with problems of phasing and missing data recovery on family trios data. Formally, given a set of genotypes partitioned into family trios, the Trio Phasing Problem (TPP) requires to find for each trio a quartet of parent haplotypes which agree with all three genotypes. Trio Missing Data Recovery Problem (TMDRP) asks for the SNP values missed in given genotype data.

Our contributions include

- Formulating the Pure-parsimony Trio Phasing Problem(PTPP) and the Trio Missing Data Recovery Problem (TMDRP).
- Proposing two new greedy and integer linear programming (ILP) based methods solving PTPP and TMDRP.
- Extensive experimental validation of proposed methods and comparison with the previously known methods.

The rest of the paper is organized as follows. The next section summarizes the previous works on phasing and missing data recovery in family trio data. In Section 3 we describe pure-parsimony trio phasing problem and propose greedy and ILP based method to solve it. In Section 4 we present our experimental study of the suggested methods on simulated and real data.

2 Previous Work

In this section we overview previous research and on phasing based on statistical methods (Phamily and PHASE, HAPLOTYPER, HAP and greedy algorithms). The ILP based approaches will be discussed in the next section.

Stephens et al. [16] introduced a Bayesian statistical method PHASE for phasing genotype data. It exploits ideas from population genetics and coalescent theory that make phased haplotypes to be expected in natural populations. It also estimates the uncertainty associated with each phasing. The software can deal with SNP in any combination, any size of population and missing data are allowed. The drawback of this method is that it takes long time for large population

Acherman et al. [1] described the tool Phamily for phasing the trio families based on well-known phasing tool PHASE [16]. It first uses the logical method described above to infer the SNPs in the parental haplotypes. Then offspring genotypes are discarded while the parental genotypes and known haplotypes are passed to PHASE. Because the offspring genotypes are discarded, PHASE no longer can maintain parent-offspring trio constraints resulting in 8.02% error rate for phasing Daly et al [4] data.

Niu et al [15] proposed a new Monte Carlo approach HAPLOTYPER for phasing genotype data. It first partition the whole haplotype into smaller segments then use the Gibbs sampler both to construct the partial haplotypes of each segment and to assemble all the segments together. This method can accurately and rapidly infer haplotypes for a large number of linked SNPs. The drawback of HAPLOTYPER is that it can not handle lengthy genotype with large population. It limits 100 SNPs and 500 population.

Halperin et al. [9] used the greedy method for phasing and missing data recovery. For each trio the author introduce four partially resolved haplotypes with the coordinates 0, 1 and ?. The values of 0 and 1 correspond to fully resolved SNPs which can be found via logical resolution from the section 1, while the ?'s corresponds to ambiguous and missing positions. The greedy algorithm iteratively finds the complete haplotype which covers the maximum possible number of partial haplotypes, removes this set of resolved partial haplotypes and continues in that manner. The authors replace each genotype in Daly et al [4] data with a pair of logically partial resolved haplotypes referring to each ambiguous SNP value as a ?. The ?'s constitute 16% of all data. Then extra 10% of data are erased (i.e., replaced with ?'s) and the resulted 26% of ambiguous SNP values are inferred by the greedy algorithm minimizing haplotype variability within blocks. When measured on the additionally erased 10% of data, the error rate for the greedy algorithm is 2.8% [9] which has been independently confirmed in our computational experiments. Unfortunately, the error rate o for the original 16% of ?'s is at least 25% which has been measured by the number of inconsistently phased SNPs. This may lead to a conclusion that the complexity of missing genotype data is considerably higher than the complexity of the successfully genotyped data.

3 Pure-Parsimony Trio Phasing

Note that it is easy to find a feasible solution to TPP but the number of feasible solutions is exponential and it is necessary to choose a criteria for comparing such solutions. In [13] for haplotyping pedigree data, the objective is to minimize recombinations. That objective is not suitable for TPP since the trios are not full-fledged pedigree data and contain no clues to evidence recombination reconstruction. Thus, following [3, 6], we have decided to pursue parsimonious objective, i.e., minimization of the total number of haplotypes.

The drawback of pure parsimony is that when the number of SNPs becomes large (as well as the number of recombinations), then the quality of pure parsimony phasing is diminishing [6]. Therefore, following the approach in [7], we suggest to partition the genotypes into blocks, i.e., substrings of bounded length, and find solution for the pure parsimony problem for each block separately. Note that in case of family trios we have great advantage over the method of [7] since we do not need to solve the problem of joining blocks. Indeed, for each family trio we can make four haplotype templates (partially resolved by logic means of haplotypes) that imply unique way of gluing together blocks to arrange complete haplotypes for the entire sequence of SNPs.

3.1 Problem Formulation

Formally, let *genotype* be a vector with m coordinates each corresponding to an SNP and having one of the following values: 0 (homozygote with major allele), 1 (homozygote with minor allele), 2 (heterozygote), or ? (missing SNP value). Let *haplotype* be a vector with m coordinates where each coordinate is either 0 or 1. We say that two haplotypes *explain* a genotype if

- for any 0 (resp. 1) in the genotype vector, the corresponding coordinates in the both haplotype vectors are 0's (resp. 1's),
- for any 2 in the genotype vector, the corresponding coordinates in the two haplotype vectors are 0 and 1,
- for any ? in the genotype vector, the corresponding coordinates in the haplotypes are unconstrained (can be arbitrary).

We say that four haplotypes h_1, h_2, h_3, h_4 *explain* a family trio of genotypes (f, m, k), if h_1 and h_2 explain the genotype f, h_3 and h_4 explain the genotype m, and h_1 and h_3 explain the genotype k.

Pure-Parsimony Trio Phasing (PPTP)**.** Given $3n$ genotypes corresponding to n family trios find minimum number of distinct haplotypes explaining all trios.

3.2 Greedy Method for Trio Phasing

We apply the greedy algorithm from Halperin [9] for trio phasing. For each trio we introduce four partial haplotypes with the coordinates 0, 1 and ?. The

values of 0 and 1 correspond to fully resolved SNPs which can be found via logical resolution from the section 1, while the ?'s corresponds to ambiguous and missing positions. The greedy algorithm iteratively finds the complete haplotype which covers the maximum possible number of partial haplotypes, removes this set of resolved partial haplotypes and continues in that manner. The drawback of this greed method is to introduce error to trio constraint, even for phasing with error O(m) to the maximum concentration. In the future, we will try to modify the greedy algorithm to overcome its shortcoming.

3.3 Integer Linear Program for Trio Phasing

We have implemented the following integer linear program (ILP) formulation (1)-(4) for pure-parsimony trio phasing. It uses 0-1 variable x_i for each possible haplotype with the minimization objective:

$$Minimize \sum x_i \tag{1}$$

For each trio we introduce four template haplotypes, i.e., haplotypes with the coordinates 0,1,2 and ?: 0's and 1's correspond to fully resolved SNPs, 2's come in pairs corresponding to the genotype 2's and ?'s correspond to unconstrained SNPs. For each 2 in each template we introduce a 0-1-variable y and a constraint binding it with the variable y' corresponding to the complimentary 2:

$$y + y' = 1 \tag{2}$$

Instead of completely resolving templates, we can resolve only 2's. Then several haplotypes can fit partially resolved templates and at least one of the corresponding x-variables should be set to 1. This results in the following constraint: For any y-assignment of 2's in each template T,

$$\sum_{x \text{ fits all } y\text{'s in template } T} x \geq 1 + \sum_{i \in I_1} y_i - |I_1| + \sum_{i \in I_2} (1 - y_i) - |I_2| \tag{3}$$

We should guarantee that each template is resolved. This is done by the following constraint: For each template T,

$$\sum_{x \text{ fits template } T} x \geq 1 \tag{4}$$

4 Experimental Results

In this section we compare our greedy and ILP based methods suggested in Section 3 with previously known phasing methods such as Phamily [1], PHASE[16] and HAPLOTYPER[15] applied to phasing and missing recovery on family trio data. We first describe the test data sets then give experimental results of five methods for phasing and then for missing data recovery of family trio data.

4.1 Test Data Sets

Our algorithms are evaluated on real and simulated data. The data set collected by Daly et al. [4] is derived from the 616 kilobase region of human Chromosome $5q31$. Another real data set is collect by Gabriel et al. [5]. This data consists of genotypes of SNPs from 62 region. This data set contains about 10% missing data.

The simulated data is generated using ms [10], a well-known haplotype generator based on the coalescent model of SNP sequence evolution. The ms generator emits a haplotype population for the given number of haplotypes, number of SNPs, and the recombination rate. We have simulated Daly et al. [4] data by generating 258 populations, each population with 100 individuals and each haplotype with 103 SNPs, then randomly choosing one haplotype from each population. We only simulate parents's haplotypes, then we obtain family trio haplotypes and genotypes by random matching the parental haplotypes.

4.2 Family Trio Phasing Validation

It is clear how to validate a phasing method on simulated data since the underlying haplotypes are known. The validation on real data is usually performed on the trio data. E.g., a phasing method is applied to parents (respectively, to offspring) genotypes and the resulted haplotypes are validated on offspring's (respectively, on parents') genotypes. Unfortunately, in our case, one can not apply such validation since a trio phasing method may rely on both offspring and parents' genotypes. Therefore, we suggest to validate trio phasing by erasing randomly chosen SNP values and recording the errors in the erased SNP sites. In Tables 1, 2, 3, each row corresponds to an instance of real data (Daly

Table 1. The results for five phasing methods on the real data sets of Daly et al.[4] and Gabrile et al. [5] and on simulated data. The second column corresponds to the ratio of erased data. The C corresponds to the error of offspring. The P corresponds to the error of parents. The T corresponds to the total error

		ILP			Greedy			Phamily			PHASE			HAPLOTYPER		
Data	E	C	P	T	C	P	T	C	P	T	C	P	T	C	P	T
Daly et al. [4]	0	0.0	0.0	0.0	4.9	16.2	3.8	1.3	0.0	0.7	1.1	0.0	0.6	2.2	0.0	1.2
	1	0.2	0.5	0.2	4.8	16.8	3.8	1.2	1.4	0.7	1.3	0.2	0.7	2.1	1.0	1.6
	2	0.3	0.7	0.4	5.0	16.9	4.0	1.3	1.8	0.9	1.3	0.5	0.8	2.2	2.3	1.7
	5	0.8	2.6	1.2	5.3	17.1	4.0	1.3	1.0	1.0	1.6	0.9	1.0	2.3	7.0	2.9
	10	1.8	6.7	3.0	5.9	17.2	4.7	1.5	2.2	1.3	1.5	1.9	1.2	2.6	9.8	4.1
Gabriel et al. [5]	0	0.0	0.0	0.0	2.9	11.5	2.2	3.0	0.0	2.0	2.2	0.0	1.3	4.4	0.0	2.7
	1	0.2	0.6	0.2	2.9	12.1	2.3	3.1	0.2	2.0	2.8	0.2	1.7	4.6	1.7	1.5
	2	0.3	1.2	0.5	3.2	12.2	2.4	3.3	0.4	2.1	2.9	0.6	1.8	4.9	3.1	1.6
	5	0.8	3.4	1.1	3.4	12.2	2.9	3.4	1.3	2.5	3.0	1.4	1.6	5.4	6.3	2.1
	10	1.5	6.2	1.5	4.3	12.4	3.7	3.9	2.4	2.5	3.3	3.1	2.1	6.1	15.7	6.3
ms [10]	0	0.0	0.0	0.0	2.6	13.2	1.9	9.4	0.0	4.7	5.6	0.0	6.5	8.1	0.0	5.4
	1	0.3	1.0	0.4	2.9	13.5	1.9	10.1	0.8	4.3	5.8	1.2	5.4	8.4	2.2	5.6
	2	0.5	1.9	0.7	3.1	13.7	2.1	10.4	1.8	7.8	5.9	2.3	5.5	8.9	4.3	6.0
	5	1.3	3.8	1.9	4.3	13.9	3.1	10.6	3.8	7.6	6.1	4.7	5.9	9.2	10.2	7.0
	10	2.5	7.7	3.6	5.3	14.0	4.4	11.9	9.5	9.2	6.9	10.5	6.0	11.5	17.1	8.0

Table 2. The results for five phasing methods on simulated data sets. The column E represents the percent of erased data. The C corresponds to the true error of offspring. The P corresponds to the true error of parents. The T corresponds to the true total error

		ILP			Greedy			Phamily			PHASE			HAPLOTYPER		
Data	E	C	P	T	C	P	T	C	P	T	C	P	T	C	P	T
ms [10]	0	1.2	1.3	1.3	1.4	1.4	1.4	2.1	2.2	2.2	3.3	3.2	3.2	2.9	2.7	2.8
	1	1.3	1.3	1.3	1.3	1.4	1.4	4.5	4.0	4.3	3.2	3.3	3.2	3.0	3.2	3.1
	2	1.5	1.6	1.6	1.6	1.6	1.6	4.4	4.3	4.4	3.4	3.3	3.4	3.2	3.3	3.3
	5	2.2	2.5	2.4	2.1	2.3	2.2	4.3	4.2	4.3	3.6	3.5	3.5	3.4	3.7	3.6
	10	3.0	3.7	3.5	3.3	3.3	3.3	5.2	5.2	5.2	3.1	3.0	3.0	3.9	4.2	4.1

Table 3. The results for missing data recovery on the real and simulated data sets with five methods. The second column corresponds to the ratio of erased data. The C* corresponds to the error of offspring. The P* corresponds to the error of parents. The T* corresponds to the total error

		ILP			Greedy			Phamily			PHASE			HAPLOTYPER		
Data	E	C*	P*	T*	C*	P*	T*	C*	P*	T*	C*	P*	T*	C*	P*	T*
Daly et al. [4]	1	2.3	7.8	5.7	3.9	6.0	5.2	0.3	2.3	1.5	0.3	3.1	2.0	1.9	26.1	16.7
	2	3.1	8.6	6.5	4.0	6.0	5.2	0.2	4.7	3.0	0.2	3.7	2.4	1.7	24.5	15.9
	5	3.9	9.9	7.8	4.5	4.8	4.7	0.2	3.6	2.5	0.1	3.4	2.3	1.3	20.5	13.9
	10	5.7	13.5	10.8	4.6	5.8	5.4	0.6	4.4	3.1	0.5	4.0	2.8	1.5	21.8	14.8
Gabriel et al. [5]	1	7.7	8.0	7.9	5.6	6.4	6.1	0	2.5	1.6	0.4	3.1	2.1	1.6	21.8	14.5
	2	7.1	8.6	8.1	4.9	5.7	5.5	0	2.8	1.9	0.5	3.1	2.2	1.0	20.7	14.1
	5	7.9	8.7	8.4	5.6	5.8	5.7	0	2.3	1.5	0.1	3.3	2.2	2.5	20.7	14.6
	10	7.4	9.5	8.8	6.1	6.6	6.5	0.1	2.1	1.5	0.3	3.1	2.1	2.3	25.1	17.5
ms [10]	1	10.9	13.3	12.4	11.5	9.2	10.1	1.0	16.0	10.2	0.7	15.2	9.6	4.3	26.4	17.9
	2	11.4	12.3	11.9	11.2	8.6	9.6	1.7	15.3	10.3	0.3	15.6	10.0	4.6	20.6	14.7
	5	13.1	12.1	12.4	12.3	7.8	9.3	0.9	14.8	10.0	0.7	14.9	10.0	3.6	23.1	16.4
	10	12.0	12.4	12.3	11.6	8.9	9.8	2.3	14.4	10.3	0.7	13.9	9.3	3.4	21.9	15.5

et al. or Gabriel et al.) or simulated data (ms) and the column (E) shows the percent of erased data (0% - no data erased, 1%-10% - percent of SNP values erased).

The value of phasing errors is measured by the Hamming distance from the method's solution to the closest feasible phasing. In Tables 1 and 2, for parents (P) we report the percent of SNP values that should be inverted out of the total number of SNP values that should be inferred (i.e., number of 2 plus number of unknown values). For offspring (C), we report the percent of SNP which should be inverted with respect to the total number of SNPs. The total number of errors (T) is the percent of SNP's that should be inverted in order to obtain a feasible phasing solution.

In Table 2, we also report true error for phasing simulated genotype data which is the Hamming distance between inferred and actual simulated underlying haplotypes for offspring (C), for parents (P) and the total error (T).

4.3 Missing Data Recovery in Family Trios

Table 3 compares five methods (ILP, Greedy, Phamily, PHASE and HAPLOTYPER) on trio missing data recovery on the real data sets (Daly [4] and Gabriel [5]) and simulated data. We erase random data in trio genotypes with certain amount($1\%, 2\%, 5\%$ and 10%) of the entire data. We report the error as the number of incorrectly recovered erased positions of the genotypes on offspring (C*), parents (P*) and trios (T*) divided the total number of erased positions in parent genotypes in percentage. We count only half error if the compared paired SNP is 2 and 0 (or 1).

5 Conclusions and Future Work

In this paper we have shown that existing methods for phasing and missing data recovery for the data representing family trios are not satisfactory. We have formulated the pure-parsimony trio phasing and the trio missing data recovery problems, proposed two new greedy and integer linear programming based solution methods, and experimentally validated of proposed methods showing advantage over the previously known methods. It has been shown that the simple greedy algorithm is very stable in missing data recovery while the ILP method is superior in trio data phasing. Our future work will include design of new methods which will combine the feasibility and stability of ILP and greedy methods.

References

1. Ackerman, H. et al (2003) 'Haplotypic analysis of the TNF locus by association efficiency and entropy', *Genome Biology*, 4:R24.
2. Brown, D.G. and Harrower, I.M. (2004) 'A new integer programming formulation for the pure parsimony problem in the haplotype association', *Workshop on Algorithms in Bioinformatics*, v.3240, 3-540-23018-1
3. Clark, A. (1990) 'Inference of haplotypes from PCR-amplified samples of diploid populations', *Mol. Biol.*, Evol, 7:111–122.
4. Daly, M., Rioux, J., Schaffner, S., Hudson, T. and Lander, E. (2001) 'High resolution haplotype structure in the human genome', *Nature Genetics*, 29:229–232.
5. Gabriel, G., Schaffner, S., Nguyen, H., Moore, J., Roy, J., Blumenstiel, B., Higgins, J., DeFelice, M., Lochner, A., Faggart, M., Liu-Cordero, S., Rotimi, C., Adeyemo, A., Cooper, R., Ward, R., Lander, E., Daly, M. and Altshuler, D. (2002) 'The structure of haplotype blocks in the human genome', *Science*, 296:2225-2229.
6. Gusfield, D. (2003) 'Haplotype inference by pure parsimony', *In R. Baeza-Yates, E. Chavez, and M. Chrochemore,ed. 14'th Annual Symposium on Combinatorial Pattern Matching*, v. 2676 of Springer LNCS, 144–155.
7. Halperin, E. and Eskin, E. (2004) 'Haplotype reconstruction from genotype data using imperfect phylogeny', *Bioinformatics*, 20(12):1842-9.

8. Halperin, E. and Karp, R.M. (2003) 'Large Scale Reconstruction of Haplotypes from Genotype Data', *International Conference on Research in Computational Molecular Biology*, 104–113.
9. Halperin, E. and Karp, R.M. (2004) 'Perfect phylogeny and haplotype assignment', *International Conference on Research in Computational Molecular Biology*, 1-58113-755-9.
10. Hudson, R. (1990) 'Gene genealogies and the coalescent process', *Oxford Survey of Evolutionary Biology*, 7:1–44.
11. He, J. and Zelikovsky, A. (2004) 'Linear Reduction for Haplotype Inference', *Proc. Workshop on Algorithms in Bioinformatics*, September 2004, Lecture Notes in Bioinformatics, 3240:242-253.
12. He, J. and Zelikovsky, A. (2004) 'Linear Reduction Methods for Tag SNP Selection', *Proc. International Conf. of the IEEE Engineering in Medicine and Biology*, 2840-2843.
13. Li, J. and Jiang, J. (2003) 'Efficient Rule-Based Haplotyping Algorithm for Pedigree Data. In Proc.', *International Conference on Research in Computational Molecular Biology*, 197-206
14. Lin, S., Chakravarti, A. and Cutler, D.J. (2004) 'Haplotype and Missing Data Inference in Nuclear Families', *Genome Res*,14(8):1624-32.
15. Niu, T., Qin, Z., Xu, X. and Liu, J.S. (2002) 'Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms', *Am. J. Hum. Genet*, 70:157–169.
16. Stephens, M., Smith, N. and Donnelly, P. (2001) 'A new statistical method for haplotype reconstruction from population data', *Am. J. Human Genetics*, 68:978–989.

Highly Scalable Algorithms for Robust String Barcoding*

B. DasGupta[1], K.M. Konwar[2], I.I. Măndoiu[2], and A.A. Shvartsman[2]

[1] Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053
dasgupta@cs.uic.edu
[2] Computer Science and Engineering Department, University of Connecticut, 371 Fairfield Rd., Storrs, CT 06269-2155
{kishori, ion, aas}@cse.uconn.edu

Abstract. String barcoding is a recently introduced technique for genomic-based identification of microorganisms. In this paper we describe the engineering of highly scalable algorithms for robust string barcoding. Our methods enable distinguisher selection based on *whole genomic sequences* of hundreds of microorganisms of up to bacterial size on a well-equipped workstation, and can be easily parallelized to further extend the applicability range to thousands of bacterial size genomes. Experimental results on both randomly generated and NCBI genomic data show that whole-genome based selection results in a number of distinguishers nearly matching the information theoretic lower bounds for the problem.

1 Introduction

String barcoding is a recently introduced technique for genomic-based identification of microorganisms such as viruses or bacteria. The basic barcoding problem [14] is formulated as follows: given the genomic sequences $g_1, \ldots g_n$ of n microorganisms, find a minimum number of strings $t_1, \ldots, t_k$ *distinguishing* these genomic sequences, i.e., having the property that, for every $g_i \neq g_j$, there exists a string t_l which is a substring of g_i or g_j, but not of both. A closely related formulation was independently proposed in [3], where it is assumed that it is possible to detect not just the presence or absence of a distinguisher t_i, but also the number of repetitions of t_i as a substring, up to a threshold of $R > 0$. The formulation in [14], which we adopt in this paper, corresponds to $R = 1$.

Identification is performed by spotting or synthesizing on a microarray the Watson-Crick complements of the distinguisher strings $t_1, \ldots, t_k$, and then hybridizing to the array the fluorescently labeled DNA extracted from the unknown microorganism. Under the assumption of perfect hybridization stringency, the hybridization pattern can be viewed as a string of k zeros and ones, referred to as the *barcode* of the microorganism. By construction, the barcodes corresponding

* The work of BD was supported in part by NSF grants CCR-0206795, CCR-0208749 and NSF CAREER grant IIS-0346973. The work of KMK and AAS was supported in part by NSF ITR grant 0121277. The work of IIM was supported in part by a "Large Grant" from the University of Connecticut's Research Foundation.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1020–1028, 2005.
© Springer-Verlag Berlin Heidelberg 2005

to the n microorganisms are distinct, and thus the barcode uniquely identifies any one of them. To improve identification robustness, one may also require *redundant distinguishability* (i.e., at least m different distinguishers for every pair of microorganisms, where $m > 1$ is some fixed constant) and impose a lower bound on the edit distance between any pair of selected distinguishers [14].

The algorithms previously proposed for string barcoding are based on integer programming [14], and on Lagrangian relaxation and simulated annealing [3]. Unfortunately, the run-time of these algorithms does not scale well with the number of microorganisms and the length of the genomic sequences, e.g., the largest instance sizes reported in [14] have a total genomic sequence length of around 100,000 bases.

In this paper we describe the engineering of highly scalable algorithms for robust string barcoding. Our methods enable distinguisher selection based on *whole genomic sequences* of hundreds of microorganisms of up to bacterial size on a well-equipped workstation, and can be easily parallelized to further extend the applicability range to thousands of bacterial size genomes. Whole-genome based selection is beneficial in at least two significant ways. First, it simplifies assay design since the DNA of the unknown pathogen can be amplified using inexpensive general-purpose whole-genome amplification methods such as specialized forms of degenerate primer multiplex PCR [5] or multiple displacement amplification [9]. Second, whole-genome based selection results in a reduced number of distinguishers, often very close to the information theoretic lower bound of $\lceil \log_2 n \rceil$.

Our algorithms are based on a simple greedy selection strategy – in every iteration we pick a substring that distinguishes the largest number of not-yet-distinguished pairs of genomic sequences. This selection strategy is an embodiment of the greedy setcover algorithm (see, e.g., [16]) for a problem instance with $O(n^2)$ elements corresponding to the pairs of sequences. Hence, by a classical result of [6, 11, 13], our algorithm guarantees an approximation factor of $2 \ln n$ for the barcoding problem. Very recently, Berman et al. [1] have shown that no approximation algorithm can guarantee a factor of $(1 - \epsilon) \ln n$ unless $NP = DTIME(n^{\log \log n})$, and also proposed an information content greedy heuristic achieving an approximation factor of $1 + \ln n$. Experimental results given in Section 4 show that our setcover greedy algorithm produces solutions of virtually identical quality to those obtained by the information content heuristic.

The setcover greedy algorithm is extremely versatile, and can be easily extended to handle redundancy and minimum edit distance constraints, as well as other biochemical constraints on individual distinguisher sequences. Furthermore, unlike the information content heuristic of [1], the greedy setcover algorithm can also take into account genomic sequence uncertainties expressed in the form of degenerate bases. Although degenerate bases are ubiquitous in genomic databases, previous works have not recognized the need to properly handle them. For example, experiments in [14] have implicitly treated degenerate bases in the input genomic sequences as distinct nucleotides; under this approach a substring of degenerate nucleotides such as NNNNN, might be erroneously selected as a distinguisher although it encodes for any possible substring of length 5.

To achieve high scalability, our implementation relies on several techniques. First, we use an incremental algorithm for quickly generating a representative set of candidate distinguishers and collecting all their occurrences in the given

genomic sequences. To reduce the number of candidates, we avoid generating any substring that appears in all genomic sequences, which typically eliminates very short candidates. For each genomic sequence, we also generate only one of the substrings that appear exclusively in that sequence, this optimization eliminates from consideration most candidate distinguishers above a certain length. Unlike the suffix tree method proposed by Rash and Gusfield [14], our approach may generate multiple candidates that appear in the same set of k genomic sequences (for $1 < k < n$). However, the penalty of having to evaluate redundant candidates in the candidate selection phase is offset in practice by the faster candidate generation time. Finally, the efficient implementation of the greedy selection phase of algorithm combines a partition based method for computing the coverage gain of candidate distinguishers (this method was first proposed in the context of the information content heuristic in [1]) with a "lazy" strategy for updating coverage gains.

The rest of the paper is organized as follows. In Section 2 we give formal problem formulations and review previous work. In Section 3 we describe the efficient implementation of the setcover greedy algorithm for the basic string barcoding problem. In Section 4 we give experimental results on complete microbial genomic sequences extracted from NCBI databases [7].

We conclude in Section 5 with directions for further research.

2 Preliminaries and Problem Formulation

Let $\Sigma = \{a, c, g, t\}$ be the DNA alphabet, and Σ^* be the set of string over Σ. A *degenerate base* is a non-empty subset of Σ. We identify degenerate bases of cardinality 1 with the respective non-degenerate bases. Given a DNA string $x = x_1 \dots x_k \in \Sigma^*$ and a string of degenerate bases $y = y_1 \dots y_n$, $n \geq k$, we say that

- x has a *perfect match* at position i of y iff $y_{i+j-1} = \{x_j\}$ for every $1 \leq j \leq k$,
- x has a *perfect mismatch* at position i of y iff there exists $1 \leq j \leq k$ such that $\{x_j\} \not\subseteq y_{i+j-1}$, and
- x has an *uncertain match* at position i of y iff $\{x_j\} \subseteq y_{i+j-1}$ for every $1 \leq j \leq k$, but $y_{i+j-1} \neq \{x_j\}$ for at least one j.

String $x = x_1 \dots x_k \in \Sigma^*$ *distinguishes* two sequences of degenerate bases y and z iff (a) x has a perfect match at one or more positions of y, and has perfect mismatches at all positions of z, or, symmetrically, (b) x has a perfect match at one or more positions of z, and has perfect mismatches at all positions of y. The robust string barcoding problem with degenerate bases is formulated as follows: *Given sequences of degenerate bases $g_1, \dots g_n$ and redundancy threshold m, find a minimum number of strings $t_1, \dots, t_k \in \Sigma^*$ such that, for every $i \neq j$, there exist m distinct strings t_l distinguishing g_i and g_j.*

It is easy to see that, for $m = 1$, at least $\lceil \log_2 n \rceil$ distinguishers are needed to distinguish any n genomic sequences. However, achieving this lower bound requires distinguishers that have perfect matches in nearly half of the sequences. In practice, additional constraints, such as lower bounds on the length of distinguishers, may result in no string having perfect matches in a large number of

sequences, and therefore much more than a logarithmic number of distinguishers. The next theorem establishes, under a simple probabilistic model, that there is an abundance of distinguishers perfectly matching at least a constant fraction of the input sequences; see [8] for the proof.

Theorem 1. *Consider a randomly generated instance of the string barcoding problem over a fixed alphabet Σ in which there are n strings, each string $s = s_0 s_1 \ldots s_{\ell-1}$ is of length exactly ℓ generated independently randomly with $Pr[s_i = a] = 1/|\Sigma|$ for any i and any $a \in \Sigma$. Also assume that ℓ is sufficiently large compared to n. Then, for a random string $x \in \Sigma^*$ of length $O(\log \ell)$, the expected number of the input strings which contain x as a substring is pn for some constant $0 < p < 1$.*

Previous work. The robust string barcoding problem was introduced (for the case when genomic sequences contain no degenerate bases) by Rash and Gusfield [14]; they provided some experimental results based on integer programming methods, and left open the exact complexity and approximability of this problem. The problem without redundancy constraints was independently considered by Borneman *et al.* [3], who also considered non-binary distinguishability (based on detecting the multiplicity of a distinguisher as a substring) and a slightly more general problem in which the objective is to pick a given number of distinguishers maximizing the number of distinguished pairs. The main motivation for the formulations in [3] comes from minimizing the number of oligonucleotide probes needed for analyzing populations of ribosomal RNA gene (rDNA) clones by hybridization experiments on DNA microarrays. Borneman *et al.* provided computational results using Lagrangian relaxation and simulated annealing techniques, and noted that the problem is NP-hard assuming that the lengths of the sequences in the prespecified set were unrestricted. Very recently, Berman, DasGupta and Kao [1] considered a general framework for test set problems that captured the string barcoding problem and its variations; their main contribution is to establish theoretically matching lower and upper bounds on the worst-case approximation ratio. Cazalis et al. [4] have independently investigated similar greedy distinguisher selection strategies for string barcoding. Unlike our work, the algorithms in [4] consider only a small random subset of the possible distinguishers and also prescribe their length in order to achieve practical running time.

3 Efficient Implementation of the Greedy Setcover Algorithm

In this section we present the implementation of the setcover greedy algorithm in the context of the basic string barcoding problem, i.e., we disregard redundancy constraints and the presence of degenerate bases in the input sequences. Implementation modifications needed to handle the robust barcoding problem in its full generality are discussed in [8].

Our implementation of the setcover greedy algorithm has two main phases: a *candidate generation phase* and a *candidate selection phase*. In the candidate generation phase a representative set of candidate distinguishers is generated from the given genomic sequences. For each generated candidate, we also compute the

Input: Set C of candidate distinguishers
Output: Set D of selected distinguishers

$D \leftarrow \emptyset$; For every $c \in C$, $\Delta_{old}(c) \leftarrow \infty$
Repeat
  $\Delta^* \leftarrow 0$
  For every $c \in C$ with $\Delta_{old}(c) > \Delta^*$ do // Since $\Delta(c, D) \leq \Delta_{old}(c)$, c can be ignored if $\Delta_{old}(c) \leq \Delta^*$
    $\Delta_{old}(c) \leftarrow \Delta(c, D)$
    If $\Delta(c, D) > \Delta^*$ then $\Delta^* \leftarrow \Delta(c, D)$; $c^* \leftarrow c$
  If $\Delta^* > 0$ then $D \leftarrow D \cup \{c^*\}$
While $\Delta^* > 0$

Fig. 1. The greedy candidate selection algorithm

list of sequences with which the candidate has perfect matches; this information is needed in the candidate selection phase. To reduce the number of candidates, we avoid generating any substring that appears in all genomic sequences, which typically eliminates very short candidates. For each genomic sequence, we also make sure to generate only one of the substrings that appear exclusively in that sequence, this optimization eliminates from consideration most candidate distinguishers above a certain length. Unlike the suffix tree method proposed by Rash and Gusfield [14], our approach may generate multiple candidates that appear in the same set of k genomic sequences (for $1 < k < n$). However, the penalty of having to evaluate redundant candidates in the candidate selection phase is offset in practice by the faster candidate generation time.

Efficient implementation of the above candidate elimination rules is achieved by generating candidates in increasing order of length and using exact match positions for candidates of length $l - 1$ when generating candidates of length l. For each position p in the input genomic sequences, we also maintain a flag to indicate whether or not the algorithm should evaluate candidate substrings starting at p. The possible values for the flag are TRUE (the substring of current length starting at p is a possible candidate), FALSE (we have already saved the substring of current length starting at p as a candidate), or DONE (all candidates containing as prefix the substring of current length starting at p are redundant, i.e., the position can be skipped for all remaining candidate lengths). Initially all flags are set to TRUE. The FALSE flags are reset to TRUE whenever we increment the candidate length, however, we never reset DONE flags. For every candidate length l, candidate evaluation proceeds sequentially over all positions of the genomic sequences. Whenever we reach a position p whose flag is set to TRUE, we use the list of matches for the substring of length $l-1$ starting at p (or a linear time string matching algorithm if l is the minimum candidate length) to determine the list of matches for the substring of length l starting at p, and set the flag to FALSE for all positions where these matches occur. If the substring of length l starting at p has matches only within the source sequence, and we have already generated a "unique" candidate for this sequence, we discard the candidate and set the flag of p to DONE.

A further speed-up technique is to generate candidate distinguishers from a strict subset of the input sequences. Although this speed-up can potentially affect solution quality, the results in Section 4 show that the solution quality

loss for whole-genome barcoding is minimal, even when we generate candidates based on a single input sequence, which corresponds to pre-assigning a barcode of all 1's to this sequence.

After the set of candidates is generated we select the final set of distinguishers in the greedy phase of the algorithm (Figure 1). We start with an empty set of distinguishers D. While there are pairs of sequences that are not yet distinguished by D, we loop over all candidates and compute for each candidate c the number $\Delta(c, D)$ of pairs of sequences that are distinguished by c but not by D, then add the candidate c with largest Δ value to D. Two sequences s and s' are distinguished by a candidate c iff exactly one of s and s' appears in the list P_c of perfect matches of c, which is available from the candidate generation phase. A simple method for computing Δ values is to maintain an $n \times n$ symmetric matrix indicating which of the pairs of sequences are already distinguished, and then to probe the $|P_c| \cdot (n - |P_c|)$ entries in this matrix corresponding to pairs (s, s') with $s \in P_c$ and $s' \notin P_c$ when computing $\Delta(c, D)$. A more efficient method is based on maintaining the partition defined on the set of sequences by D. If the partition defined by D consists of sets $S_1, \ldots, S_k$, then we can compute $\Delta(c, D)$ in $O(k + |P_c|) = O(n)$ time using the observation that

$$\Delta(c, D) = \sum_{i=1}^{k} |S_i \cap P_c| \cdot |S_i \setminus P_c| \tag{1}$$

In addition to the fast partition based computation, our implementation of the greedy selection phase uses a lazy strategy for updating the Δ values, based on the observation that they are monotonically non-increasing during the algorithm (see Figure 1).

4 Experimental Results

We performed experiments on both randomly generated instances and whole microbial genomes extracted from the NCBI databases [7]. Due to space constraints we report here only results obtained by the setcover greedy algorithm on microbial genomes. Experimental results comparing the setcover greedy algorithm, the information content heuristic in [1], and a recently proposed multi-step rounding algorithm for set multicover [2] on large number of random testcases are given in [8].

The results on the NCBI dataset, which consists of a set of 29 complete microbial genomic sequences extracted from [7], are given in Table 1. Sequence lengths in the set vary between 490 Kbases and 4.75 Mbases, with an average length of 2.6 Mbases (over 76 Mbases total). The sequences contain a small number of degenerate bases, 861 bases in total. Therefore, we cannot use the partition method for computing the number of sequence pairs distinguished by a candidate in the greedy selection phase, and we have to use the slower matrix datastructure. In these experiments we varied the redundancy requirement from 1 to 20. To see the effect of length and edit distance requirements on the number of distinguishers, for each redundancy requirement we computed both an unconstrained solution, and a solution in which distinguishers must have length between 15 and 40, and there should be a minimum edit distance of 6 between

Table 1. Results on a set of 29 NCBI complete microbial genomes. Candidate generation time is approximately 335 seconds for all combinations of parameters. The experiments were run on a PowerEdge 2600 Linux server with 4 Gb of RAM and dual 2.8 GHz Intel Xeon CPUs – only one of which is used by our sequential algorithms

Redundancy	l_{min}	l_{max}	MinEdit	Select time	#Distinguishers
1	0	∞	0	14.2	6.0
1	15	40	6	2.6	8.0
5	0	∞	0	20.3	21.0
5	15	40	6	8.7	31.0
10	0	∞	0	22.9	41.0
10	15	40	6	16.4	60.0
20	0	∞	0	26.8	76.0
20	15	40	6	33.4	123.0

every two selected distinguishers (these values are similar to those used in [14]. In all experiments, we generated candidates based only on the shortest sequence of 490 Kbases.

Results in Table 1 show that, naturally, meeting higher redundancy constraints requires more distinguishers to be selected. Additional length and edit distance constraints further increase the number of distinguishers, but the latter is still within reasonable limits. The length constraints reduce the number of candidates (from 1,775,471 to 122,478), which, for low redundancy values has the effect of reducing greedy selection time. However, for high redundancy requirements the reduction in number of candidates is offset by the increase in solution size, and greedy selection becomes more time consuming with length and edit distance than without (selection time grows roughly linearly with solution size).

5 Conclusions

In this paper we have given highly scalable algorithms for the robust string barcoding problem, and have shown that distinguisher selection based whole genomic sequences results in a number of distinguishers nearly matching the information theoretic lower bounds for the problem.

In ongoing work we are exploring heuristics and approximation algorithms for several extensions of the string barcoding problem. First, we are considering the use of probe mixtures as distinguishers. With most microarray technologies it is feasible to spot/synthesize a mixture of oligonucleotides at any given microarray location. The DNA of a pathogen will hybridize to such a location if it contains at least one substring which is the Watson-Crick complement of one of the oligonucleotides in the mixture. Using oligonucleotide mixtures as distinguishers can reduce the number of spots on the array – and therefore barcode length – closer to the information theoretical lower-bound of $\log_2 n$. The reduction promises to be particularly significant when reliable hybridization requires relatively long distinguishers; in these cases even the optimum barcoding length is far from $\log_2 n$ [14]. A special case of this approach is the use of *degenerate* distinguishers similar to the degenerate primers that have been recently employed

in multiplex PCR amplification [12, 15]. Degenerate distinguishers are particularly attractive for string barcoding since their synthesis cost is nearly identical to the synthesis cost of a single non-degenerate distinguisher (synthesis requires the same number of steps, the only difference is that multiple nucleotides must be added in some of the synthesis steps).

In many practical pathogen identification applications collected biological samples may contain the DNA of multiple pathogens. This issue is considered to be particularly significant in medical diagnosis applications, see, e.g., [10] for studies in detecting more than one HPV (human papiloma virus) genotype with varying rate of multiple HPV infections carried by the same HPV carrier. In future work we plan to develop extensions of the barcoding technique that can reliably detect multiple pathogens for a given bound on the number of pathogens present.

References

1. P. Berman, B. DasGupta, and M.-Y. Kao. Tight approximability results for test set problems in bioinformatics. In *Proc. 9th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 3111 of *Lecture Notes in Computer Science*, pages 39–50, Berlin, 2004. Springer-Verlag.
2. P. Berman, B. DasGupta, and E. Sontag. Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene networks. In *Proc. 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 2748 of *Lecture Notes in Computer Science*, pages 39–50, Berlin, 2004. Springer-Verlag (To appear in *Discrete Applied Mathematics*.).
3. J. Borneman, M. Chrobak, G.D. Vedova, A. Figueora, and T. Jiang. Probe selection algorithms with applications in the analysis of microbial communities. *Bioinformatics*, 1:1–9, 2001.
4. D. Cazalis, T.Milledge, and G. Narasimhan. Probe selection problem: Structure and algorithms. In *Proc. 8th Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2004)*, pages 124–129, 2004.
5. V.G. Cheung and S.F. Nelson. Whole genome amplification using a degenerate oligonucleotide primer allows hundreds of genotypes to be performed on less than one nanogram of genomic dna. *Proc. Natl. Acad. Sci. USA.*, 93:14676–14679, 1996.
6. V. Chvátal. A greedy heuristic for the set covering problem. *Math. of Op. Res.*, 4:233–235, 1979.
7. NCBI Completed Microbial Genomes. http://www.ncbi.nlm.nih.gov/genomes/microbes/complete.html, 2004.
8. B. DasGupta, K. Konwar, I.I. Mandoiu, and A. Shvartsman. Highly scalable algorithms for robust string barcoding. ACM Computing Research Repository, cs.DS/0502065, 2005.
9. F.B. Dean, S. Hosono, L. Fang, X. Wu, A. Fawad Faruqi, P. Bray-Ward, Z. Sun, Q. Zong, Y. Du, J. Du, M. Driscoll, W. Song, S.F. Kingsmore, M. Egholm, and R.S. Lasken. Comprehensive human genome amplification using multiple displacement amplification. *Proc. Natl. Acad. Sci. USA.*, 99:5261–5266, 2002.
10. B. Gharizadeh, M. Käller, P. Nyrén, A. Andersson, M. Uhlén, J. Lundeberg, and A. Ahmadian. Viral and microbial genotyping by a combination of multiplex competitive hybridization and specific extension followed by hybridization to generic tag arrays. *Nucleic Acids Research*, 31(22), 2003.
11. D.S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Sys. Sci.*, 9:256–278, 1974.

12. C. Linhart and R. Shamir. The degenerate primer design problem. *Bioinformatics*, 18:S172–S181, 2002.
13. L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.
14. S. Rash and D. Gusfield. String barcoding: Uncovering optimal virus signatures. In *Proc. 6th Annual International Conference on Computational Biology*, pages 254–261, 2002.
15. R. Souvenir, J. Buhler, G. Stormo, and W. Zhang. Selecting degenerate multiplex PCR primers. In *Proc. 3rd Intl. Workshop on Algorithms in Bioinformatics (WABI)*, pages 512–526, 2003.
16. V.V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.

Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection

Ferdinando Cicalese[1,⋆], Peter Damaschke[2,⋆⋆] , and Ugo Vaccaro[3]

[1] Institut für Bioinformatik, Centrum für Biotechnologie (CeBiTec), Universität Bielefeld, 33594 Bielefeld, Germany
nando@cebitec.uni-bielefeld.de
[2] School of Computer Science and Engineering, Chalmers University, 41296 Göteborg, Sweden
ptr@cs.chalmers.se
[3] Università di Salerno, 84081 Baronissi (SA), Italy
uv@dia.unisa.it

Abstract. The classical Group Testing Problem is: Given a finite set of items $\{1, 2, \ldots, n\}$ and an unknown subset $P \subseteq \{1, 2, \ldots, n\}$ of up to p positive elements, identify P by asking the least number of queries of the type "does the subset $Q \subseteq \{1, 2, \ldots, n\}$ intersect P?". In our case, Q must be a subset of *consecutive* elements. This problem naturally arises in several scenarios, most notably in Computational Biology. We focus on algorithms in which queries are arranged in *stages*: in each stage, queries can be performed in parallel, and be chosen depending on the answers to queries in previous stages. Algorithms that operate in few stages are usually preferred in practice. First we study the case $p = 1$ comprehensively. For two-stage strategies for arbitrary p we obtain asymptotically tight bounds on the number of queries. Furthermore we prove bounds for any number of stages and positives, and we discuss the problem with the restriction that query intervals have some bounded length d.

1 Introduction and Contributions

In group testing, the task is to determine the "positive" members of a set of objects O by asking as few queries as possible of the form "does the subset $Q \subseteq O$ contain at least one positive object?". A negative answer to a query tells us that all items in Q are "negative". Group testing is a paradigm occuring in a variety of situations such as quality control, multiple access communication, computational molecular biology, and data compression, among the others (see [5, 6, 9, 7, 2, 10, 12] and the numerous references quoted therein).

⋆ Supported by DAAD-grant no. A/04/33535.
⋆⋆ Supported by the Swedish Research Council (Vetenskapsrådet), project "Algorithms for searching and inference in genetics", grant no. 621-2002-4574.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1029–1037, 2005.
© Springer-Verlag Berlin Heidelberg 2005

In this paper we consider a variant in which the set of objects is $O = \{1, 2, \ldots, n\}$, the unknown subset $P \subseteq O$ of positive elements has cardinality at most p, and queries (tests) are constrained to be intervals $\{i, i+1, \ldots, j-1, j\}$, for some $i, j \in \{1, 2, \ldots, n\}$, such that $j - i + 1 \leq d$, d being a parameter of the problem. This variant naturally arises in several important situations. One of them concerns the determination of exon-intron boundaries within a gene [11, 13]. Adopting a very simplified model, one can view a gene as consisting of several disjoint substrings within a long string representing the DNA molecule. These substrings are called *exons*, and the substrings separating them are called *introns*. Only the concatenation of exons codes for a protein, the biological role of introns is rather unclear. Each boundary point linking an exon and an intron is called a *splice site*. The determination of splice sites is often a critical point to search for mutations associated with a gene responsible for a disease, because only mutations in exons are relevant. Now, in laboratories it is possible to obtain a "purified" version of a gene by transcribing it to cDNA in which all introns are removed and the exons spliced together. Using standard experimental procedures (polymerase chain reaction, PCR) one can pick any two positions in the cDNA string and determine whether they are at the same distance as they were in the original genomic DNA string. If these distances do not coincide then at least one intron (and hence a splice site) must be present in the genomic DNA between the two picked positions. The formulation of splice sites identification as a group testing problem with interval queries is explicitly stated in [8, 11, 13], where parameter d takes into account technological limitations of the PCR procedures.[4] A group testing procedure actually finds the positions (base or group of bases) before (or after) which there was an intron that has been spliced.

In the area of searching two kinds of algorithms are usually considered: In *adaptive* algorithms the tests are performed one by one, and the outcome of previous tests are assumed known at the time of determining the current test. Conversely, in *non-adaptive* algorithms all tests must be specified in advance without knowing the outcomes of the other tests. An intermediate situation corresponds to algorithms in which tests can be arranged in *stages*: in each stage a certain number of tests is performed non-adaptively, while tests of a given stage can be determined depending on the outcomes of the tests in the previous stages. In many situations, for instance in the biological setting, "few"-stage procedures are by far the most preferable ones [6, 9]. According to [9] *... the technicians who implement the pooling strategies generally dislike even the* 3*-stage strategies that are often used [...]. The pools are either tested all at once or in a small number of stages (usually at most 2).* In this paper we shall concentrate on s-stage group testing procedures, particularly for $s = 1$ and $s = 2$.

[4] In [13] it is mentioned that current technology suggests also a lower bound on the test size. One can see (as also noticed in [13]) that in the worst case this additional constraint makes impossible the complete classification of each item as either positive or negative.

To the best of our knowledge our paper is the first to provide a rigorous algorithmic analysis of group testing with interval queries. In particular, it is interesting to know that a simple strategy for any number of positives is provably optimal in the 2-stage case.

The work [13] and the book [11] report about the experimental evaluation, on real data, of the algorithm ExonPCR, that finds exon-intron boundaries within a gene. The authors of [13] give also a simple asymptotic analysis of their $\Theta(\log n)$-stage algorithm, for the case in which no bound on the test size is assumed.

Due to the space limitation, almost all proofs are removed from this extended abstract. The interested reader is referred to the full paper [1] where other scenarios in which our group testing problem naturally arises are also described.

2 Non-adaptive Interval Group Testing Algorithms

Let us denote by $\mathcal{N}_s(n,p,d)$ the worst-case minimum number of tests that are necessary (and sufficient) to successfully identify all positives in a search space of cardinality n, under the hypothesis that the number of positives is at most p, the interval tests have size at most d, and s-stage algorithms are used.

Theorem 1. *For all non-negative integers n, d, p, it holds that*

$$\mathcal{N}_1(n,p,d) = \begin{cases} 2\lfloor n/3 \rfloor + (n \bmod 3) & \text{if } p=1 \text{ and } d=2, \\ \lceil (n+1)/2 \rceil & \text{if } p=1 \text{ and } d \geq 3, \\ n & \text{otherwise.} \end{cases} \tag{1}$$

Proof. The main argument for the lower bounds is that any strategy must perform at least one test discriminating between any pair of consecutive items.

The matching upper bound $\mathcal{N}_1(n,1,3) \leq \lceil (n+1)/2 \rceil$, is given by the following strategy:

For $i = 1, 2, \ldots, \lceil (n+1)/2 \rceil$, perform the test T_i defined by

$$T_i = \begin{cases} \{1,2\} & i = 1, n > 1, \\ \{2i-2, 2i-1, 2i\} & i = 2, 3, \ldots, \lceil (n+1)/2 \rceil - 1, \\ \{n-1, n\} & i = \lceil (n+1)/2 \rceil \text{ and } n \text{ is odd}, \\ \{n\} & i = \lceil (n+1)/2 \rceil \text{ and } n \text{ is even, or } n = 1. \end{cases}$$

It is easy to check that this strategy defines a one-to-one mapping from the set of all possible solutions to the set of tests' outcomes.

The other cases are similar. □

3 Multistage Interval Group Testing

We shall now focus on s-stage interval group testing algorithms consisting of a fixed number s of stages. In each stage tests are performed in parallel so that the outcomes of the tests are taken into account only for choices made in the following stages.

For any integer $s \geq 2$, an s-stage interval group testing algorithm $\mathcal{A}$ consists of s successive 1-stage algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_s$, confining the positives in a collection of smaller and smaller subintervals. The last stage returns subintervals of size one since it determines the exact positions of the positives. Our main interest will be on case $s = 2$.

3.1 At Most One Positive

We shall first consider s-stage algorithms for the special case when at most one positive item is present in the search space. The main result of this section is contained in Theorem 2.

Proposition 1. *Let $I = \{1, 2, \ldots, n\}$ contain at most one positive item. Let $\mathcal{A}$ be a set of q interval tests in I such that for all possible choices of the positive item (including the choice for which no item in I is positive), the outcomes of the tests of $\mathcal{A}$ reduce the space of items in I which are candidate to be the positive one to a set of cardinality at most a. Then $q \geq \lceil n/(2a) \rceil$.*

In the following, an interval group testing algorithm *for at most one positive* in a set I is an algorithm that either finds the positive in I or correctly decides that no positive is in I. An interval group testing algorithm *for exactly one positive* in I is an algorithm that finds the positive element in a set I already known to contain a positive. These two notions are related by an obvious statement:

Proposition 2. *Let I be a set of cardinality n and I' be a set of cardinality $n+1$. For any interval group testing algorithm $\mathcal{A}$ for at most one positive in I, there exists an interval group testing algorithm $\mathcal{B}$ for exactly one positive, in I', which performs the same number of tests as $\mathcal{A}$.*

Trivially $\mathcal{B}$ can run the algorithm $\mathcal{A}$ on the first n elements of I'. If $\mathcal{A}$ finds a positive then $\mathcal{B}$'s output will coincide with $\mathcal{A}$'s output, otherwise $\mathcal{B}$ will conclude that the positive is the $n+1$st element of I'.

The following lemma is the key tool for analyzing s-stage interval group testing algorithms for one positive.

Lemma 1. *Fix integers $s \geq 1$ and $q \geq s$. Let $t = \lfloor q/s \rfloor$ and $k = q \bmod s$, and let $n = 2^s t^{s-k}(t+1)^k - 1$. There exists an s-stage interval group testing algorithm $\mathcal{A}$ for at most one positive element in $I = \{1, 2, \ldots, n\}$, which performs q tests. Moreover, there exists no s-stage interval group testing algorithm $\mathcal{A}$ for at most one positive element in $I = \{1, 2, \ldots, n+1\}$ which performs q tests*

Proof. The existence part is by induction on s. For $s = 1$ and for all $q \geq 1$ we have $k = 0$, $t = q$ and $n = 2q - 1$. Thus the desired result directly follows by Theorem 1.

Let now $s \geq 2$, $q \geq s$ and $t = \lfloor q/s \rfloor$, $k = q \bmod s$. Let $n = 2^s t^{s-k}(t+1)^k - 1$ and let $I = \{1, 2, \ldots, n\}$.

Assume as an induction hypothesis that for all $q' \geq s - 1$ and for $t' = \lfloor q'/(s-1) \rfloor$, $k' = q' \bmod (s-1)$ and $\hat{n} = 2^{s-1} t'^{s-1-k'}(t'+1)^{k'}$ there exists an

$(s-1)$-stage algorithm for at most one positive in a set of cardinality $\hat{n}-1$ which uses q' tests. Notice that by Proposition 2 this implies that there exists an $(s-1)$-stage algorithm for exactly one positive in a set of cardinality $\hat{n}$ which uses q' tests.

An algorithm $\mathcal{A}$ that achieves the claimed result is as follows. In the first stage $\mathcal{A}$ uses t queries to partition the search space into $2t-1$ intervals of size $u = 2^{s-1}t^{s-k-1}(t+1)^k$ leaving an additional interval of size $u-1$ uncovered. This can be easily achieved by distributing the queries as described in the first part of Theorem 1.

By induction hypothesis and Proposition 2 the remaining $q-t$ queries suffice to complete the search in one of the intervals of size u if at least one query has answered YES and then a positive is know to be present or in the only subinterval of size $u-1$ if no test has answered YES.

The second statement of the lemma is proved by contradiction (omitted). □

Theorem 2. *Fix an integer $s \geq 1$. For all integers $n \geq 1$ we have*

$$\mathcal{N}_s(n,1,n) = \begin{cases} \lceil \log n \rceil & \text{if } n < 2^s - 1. \\ \min\left\{ q \mid 2^s \left(\left\lfloor \frac{q}{s} \right\rfloor\right)^{s-(q \bmod s)} \left(\left\lceil \frac{q}{s} \right\rceil\right)^{q \bmod s} - 1 \geq n \right\} & \text{otherwise} \end{cases} \tag{2}$$

In particular, for $n > 2^s - 1$, we have $\mathcal{N}_s(n,1,n) = \frac{s}{2} n^{1/s} + O(s)$.

Proof. For $n < 2^s - 1$, the problem reduces to binary search. For $n \geq 2^s - 1$, the result follows straightforwardly by Lemma 1. □

3.2 More Than One Positive: Two-Stage Interval Group Testing

The aim of this section is to prove asymptotically tight upper and lower bounds on the query number of 2-stage interval group testing algorithms for *at most p positives*, for any fixed number p. Remember that p is typically much smaller than n.

Unlike the case $p = 1$, for $p \geq 2$ the possible answers to the first questions in the first stage do no longer partition the search space into mutually disjoint candidate subsets to contain the positive(s). The instantaneous description of the searcher's knowledge after one stage is considerably more complicated. Nonetheless an upper bound for the case $s = 2$ and $p \geq 2$ can be easily obtained.

Theorem 3. *For any $p \geq 2$ we have $\mathcal{N}_2(n,p,n) \leq 2\sqrt{p-1}\sqrt{n} + O(p)$.*

Proof. An obvious algorithm $\mathcal{A}$ achieves the upper bound. The t_1 queries in $\mathcal{A}_1$ split the search space in t_1 disjoint intervals of roughly equal length n/t_1. This causes $n(p-1)/t_1 + O(p)$ queries in the second stage $\mathcal{A}_2$: If p intervals say YES, there is one positive in each of them, hence $p(n/t_1)/2 + O(p) = np/2t_1 + O(p)$ queries are needed. If $p-1$ intervals say YES, each of them could contain both

positives. (Actually only one interval can, but the searcher does not see which.) Hence $\mathcal{A}_2$ needs $n(p-1)/t_1 + O(p)$ queries, which is obviously the worst case for this algorithm. The $O(p)$ term accounts for the fact that n/t_1 is in general not integer, so that every YES interval could be longer by up to 1. Choosing $t_1 = \sqrt{p-1}\sqrt{n}$ minimizes $t_1 + n(p-1)/t_1$. Consequently, $2\sqrt{p-1}\sqrt{n} + O(p)$ queries are sufficient. □

The challenge is in fact to prove that one cannot do essentially better than this obvious strategy. Assume we could show that an adversary can enforce at least $n(p-1)/t_1$ queries in the second stage for any set of t_1 intervals in the first stage. Clearly, this would imply a lower bound that misses the upper bound of Theorem 3 only by an additive $O(p)$ term. For $p = o(\sqrt{n})$ this is asymptotically tight.

We remark that, for given p, it suffices to consider case $t_1 = \omega(p)$, because if $t_1 = O(p)$, the query intervals cut the search space in $O(p)$ pieces. (Here, a piece means a maximal contiguous sequence of elements bounded by interval ends, and without further interval ends in between. The length of a piece is the number of elements in it.) Hence the p longest pieces have total length $\Omega(n)$. If the adversary places the positives in these pieces, $\Omega(n)$ queries are enforced in the second stage.

Actually, our general lower bound is slightly weaker than mentioned above. In certain cases we get only $n(p-1)/(t_1 + p/2 + 1)$, but this means merely a factor $1 + p/2t_1$ between upper and lower bound, and for $t_1 \approx \sqrt{p}\sqrt{n}$ this is $1 + \sqrt{p}/2\sqrt{n}$. Hence our final result is still asymptotically tight. The exact $\mathcal{N}_2(n, p, n)$ for any n and p remains an open problem.

We start with some notations and give a rough idea of our proof of the lower bound.

Let $\mathcal{Q}$ be a set of interval questions. As said above, the interval ends in $\mathcal{Q}$ cut the search space $\{1, 2, \ldots, n\}$ into *pieces*. For any piece a, we denote by $N^{\mathcal{Q}}(a)$ the set of query intervals in $\mathcal{Q}$ containing a.

Let $\pi_1, \ldots, \pi_\ell$ be the pieces determined by the intervals of $\mathcal{Q}$. Let x_i be the length of π_i. By a YES *set* for $\mathcal{Q}$ we understand a set of query intervals in $\mathcal{Q}$ that have answered YES. Given a YES set Y, we define the YES vector $\mathbf{w}^Y = (w_1, \ldots, w_\ell)$, where w_i is the *weight* assigned to the piece π_i's according to the following scheme:

- A piece gets weight 1/2 if it can contain a positive but not more than one.
- A piece gets weight 1 if it can contain more than one positive.

We denote with $w^Y(\mathcal{Q})$ the weighted sum of the lengths of the pieces cut by $\mathcal{Q}$ weighted according to the YES vector associated to Y. In formulas $w^Y(\mathcal{Q}) = \sum_{j=1}^{\ell} x_j w_j^Y$.

Assume now that $\mathcal{Q}$ is the set of interval questions asked in the first stage of a two stage group testing algorithm which finds more than one positive. By Theorem 1, if Y is the set of intervals in $\mathcal{Q}$ that answer YES then the number

of queries to be asked in the second stage in order to find all the positives is *at least* $w^Y(\mathcal{Q})$. Therefore, in order to prove the promised bound we show that for each possible set of interval questions $\mathcal{A}_1$ there exists a yes set Y such that $w^Y(\mathcal{A}_1) \geq n/|\mathcal{A}_1|$.

Let $\mathcal{A}_1$ be the set of questions asked in the first stage of a two stage interval group testing algorithm for p positives in the set $\{1, 2, \ldots, n\}$. Let ℓ be the number of pieces in which the search space is cut by the questions in $\mathcal{A}_1$. Let $x_1, \ldots, x_\ell$ be the length of these pieces. For each piece a we shall write $N(a)$ instead of $N^{\mathcal{A}_1}(a)$, tacitly assuming that $\mathcal{A}_1$ is the set of interval questions we are referring to.

W.l.o.g. we can assume that for each two pieces a and b determined by $\mathcal{A}_1$ it holds that $N(a) \neq N(b)$. Then, we also have that ℓ, the total number of pieces, is at most $2t_1$. In fact, the number of pieces covered by query intervals is at most $2t_1 - 1$ (this is trivial for $t_1 = 1$, and every new interval can create at most two new pieces by splitting) and w.l.o.g. at most one piece a is outside all query intervals ($N(a) = \emptyset$).

The next lemma, basically a duality argument, is the key to the lower bound.

Lemma 2. *Consider a multiset of k (not necessarily distinct!)* YES *sets, and for each $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, \ell$, let w_{ij} be the weight of the jth piece in the* YES *vector associated to the ith* YES *set. If there exist $r > 0$ such that for all $j = 1, 2, \ldots, \ell$, it holds that $\sum_{i=1}^{k} w_{ij} \geq r$, then an adversary can force at least $\frac{r}{k}n$ queries in $\mathcal{A}_2$.*

Our lower bound consists in proving that it is possible to select a multiset of YES sets such that for each piece cut by $\mathcal{A}_1$ the sum of the weights assigned by the YES sets to that piece is greater than $p - 1$. In particular, we show that in many cases it is possible to have such a multiset of cardinality $k \leq t_1$ and in general of $k \leq t_1 + p/2 + 1$.

This is done in a series of combinatorial lemmas and case distinctions that we cannot present within the limited space. See the full paper [1]. We finally get:

Theorem 4. *For $p = o(\sqrt{n})$, the 2-stage interval group testing problem for at most p positives needs $\mathcal{N}_2(n, p, n) = 2\sqrt{p-1}\sqrt{n} + O(p)$ queries, and they are also sufficient.*

3.3 More and More Stages

We can characterize the asymptotic complexity of an optimal strategy for any fixed number s of stages. So far we do not have the precise constant of the main term as, in fact, we have in the 2-stage case, but recent ideas give hope that this gap can be closed.

Theorem 5. *Let $n \geq 2^s(p-1)$ and let $I = \{1, \ldots, n\}$ contain at most $p \geq 2$ positives. There exists an s-stage interval group testing algorithm which finds all positives in I by performing a number of tests smaller than or equal to*

$$s(p-1)^{(s-1)/s}(n - (n \bmod (p-1)))^{1/s} + s + p + (n \bmod (p-1)) - 2. \quad (3)$$

3.4 Bounded Queries

We also extend the results of the previous sections to the case of strategies using bounded interval tests whose size is bounded to be not larger than a given threshold d, and we characterized the query number of the optimal strategies in many cases of interest. We cannot state these results here (see the full paper), but a general observation is in order. Since the whole search space I of cardinality n must be covered, we are forced to use $\geq \lceil n/d \rceil$ tests. Dividing the search space into that many intervals of size d in the first stage and collecting the answers to these tests, will tell which of these intervals contain any positives. Then the algorithms have to recur, but the constraint of d on the tests' size does not count anymore. Thus in the next stage an algorithm with unbounded test size will solve the problem. This is provably optimal in many cases.

References

1. F. Cicalese, P. Damaschke, U. Vaccaro, *Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection*, Technical Report 2004-04, Universität Bielefeld, Technische Fakultät. Available at `http://www.cs.chalmers.se/~ptr/2004-04.pdf`.
2. G. Cormode, S. Muthukrishnan, *What's hot and what's not: Tracking most frequent items dynamically*, in: ACM Principles of Database Systems, 2003.
3. L. A. Cox, X. Sun, and Y. Qiu, *Optimal and Heuristic Search for a Hidden Object in one Dimension*, in: Proc. of IEEE Conf. on System, Man, and Cybernetics, pp. 1252–1256, 1994.
4. A. De Bonis, L. Gasieniec, U. Vaccaro, *Generalized Framework for Selectors with Applications in Optimal Group Testing*, in: ICALP '03, LNCS **2719**, pp. 81-96, Springer-Verlag, 2003.
5. D.Z. Du and F.K. Hwang, *Combinatorial Group Testing and its Applications*, World Scientific, Singapore, 2000.
6. M. Farach, S. Kannan, E.H. Knill, S. Muthukrishnan, *Group Testing with Sequences in Experimental Molecular Biology*, in: Proc. of Compression and Complexity of Sequences 1997, B. Carpentieri, A. De Santis, U. Vaccaro, J. Storer (Eds.), IEEE CS Press, pp. 357-367, 1997.
7. E. H. Hong and R.E. Ladner, *Group testing for image compression*, IEEE Transactions on Image Processing, **11(8)**, pp. 901-911, 2002.
8. R. Karp, *ISIT'98 Plenary Lecture Report: Variations on the theme of 'Twenty Questions'*, IEEE Information Theory Society Newsletter, vol. **49**, No.1, March 1999.
9. E. Knill, *Lower Bounds for Identifying Subset Members with Subset Queries*, in: Proceedings of Symposium on Discrete Algorithms 1995 (SODA 1995), pp. 369-377, 1995.
10. Hung Q. Ngo and Ding-Zhu Du, *A survey on combinatorial group testing algorithms with applications to DNA library screening*, in: *Discrete Mathematical Problems with Medical Applications*, DIMACS Series Discrete Math. Theoret. Computer Science, **55**, Amer. Math. Soc., pp. 171-182, 2000.

11. P.A. Pevzner, *Computational Molecular Biology, An Algorithmic Approach*, MIT Press, 2000.
12. J. Wolf, *Born again group testing: Multiaccess communications*, IEEE Trans. Information Theory, **IT-31**, pp. 185-191, 1985.
13. G. Xu, S.H. Sze, C.P. Liu, P.A. Pevzner, N. Arnheim, *Gene hunting without sequencing genomic clones: Finding exon boundaries in cDNAs*, Genomics, **47**, pp. 171-179, 1998.

Virtual Gene: A Gene Selection Algorithm for Sample Classification on Microarray Datasets

Xian Xu and Aidong Zhang

State University of New York at Buffalo, Buffalo, NY 14260, USA
{xianxu, azhang}@cse.buffalo.edu

Abstract. Gene Selection is one class of most used data analysis algorithms on microarray dataset. The goal of gene selection algorithms is to filter out a small set of informative genes that best explains experimental variations. Traditional gene selection algorithms are mostly single-gene based. Some discriminative scores are calculated and sorted for each gene. Top ranked genes are then selected as informative genes for further study. Such algorithms ignore completely correlations between genes, although such correlations is widely known. Genes interact with each other through various pathways and regulative networks. In this paper, we propose to use, instead of ignoring, such correlations for gene selection. Experiments performed on three public available datasets show promising results.

1 Introduction

Microarray experiments enable biologists to monitor expression levels of thousands of genes or ESTs simultaneously [1, 5, 12]. Short sequences of genes or ESTs tagged with fluorescent materials are printed on a glass surface. The slice is then exposed to sample solution for hybridization (base-pairing). mRNA molecules are expected to hybridize with short sequences matching part of their complement sequences. After hybridization the slice is scanned and goes through various data processing steps including image processing, quality control and normalization [4]. The resulting dataset is a two dimensional array with thousands of rows (genes) and tens of columns (experiments). Element at i^{th} row and j^{th} column in such an array is the expression level measure for gene i in experiment j. When tissue samples used in the experiments are labeled (e.g., sample is cancer tissue or normal tissue), sample classification can be performed on such dataset. New samples are classified based on their gene expression profiles.

Such dataset poses special challenge for pattern recognition algorithms. The main obstacle is the limited number of samples due to practical and financial concerns. This results in the situation where the number of features (or genes) well outnumbers the number of observations. The term "curse of dimensionality" and "peaking phenomenon" are coined in the machine learning and pattern recognition community, referring to the phenomenon that inclusion of excessive features may actually degrade the performance of a classifier if the number of training examples used to build the classifier is relatively small compared to the number of features [9]. Typical treatment is to reduce the dimensionality of feature space before classification using feature extraction and feature selection. Feature extraction algorithms create new features based on transformation and/or combination of original features while feature selection algorithms

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1038–1045, 2005.
© Springer-Verlag Berlin Heidelberg 2005

aim to select a subset of original features. Techniques like PCA and SVD have been used to create salient features [6, 7] for sample classification on microarray datasets. Feature selection, or in our case, gene selection generates a small set of informative genes, which not only leads to better classifiers, but also enables further biological investigation.

In order to find the optimal subset of features that maximizes some feature selection criterion function (we assume the higher value the criterion function, the better the feature subset), straightforward implementation would require evaluation of the criterion function for each feature subset, which is a classic NP hard problem. Various heuristics and greedy algorithms have been proposed to find sub-optimal solutions. Assuming independence between features, one attempt is to combine small feature subsets with high individual scores. This heuristic is widely used for gene selection. A class of gene selection algorithms calculates discriminative scores for individual genes and combines top ranked genes as selected gene set. We refer to this class of algorithms single gene based algorithms. Various discriminative scores have been proposed, including statistical tests (t-test, F-test) [3], non-parametric tests like TNoM [2], mutual information [15, 16], S2N ratio (signal to noise ratio) [5] , extreme value distribution [11] and SAM [13] etc. Although simple, this class of algorithms is widely used in microarray data analysis and proven to be effective and efficient.

However, the assumption of independence between genes over simplifies the complex relationship between genes. Genes are well known to interact with each other through gene regulative networks. As a matter of fact, the common assumption of cluster analysis on microarray dataset [10] is that co-regulated genes have similar expression profiles. Bø [3] proposed to calculate discriminant scores for a pair of genes instead of each individual gene. Several of recent researches on feature selection especially gene selection [8, 14, 16] took into consideration the correlation between genes explicitly by limiting redundancy in resulting gene set. Heuristically, selected genes need to first have high discriminative scores individually and secondly not correlate much with genes that have already been selected. Generic feature selection algorithms like SFFS (sequential forward floating selection), SBFS (sequential backward floating selection), etc. have also been used for selecting informative genes from microarray datasets.

In this paper, we propose a totally different approach. Instead of trying to get rid of correlation in the selected gene set, we examine whether such correlation itself is a good predictor of sample class labels. Our algorithm is a supervised feature extraction algorithm based on new feature "virtual gene". "Virtual genes" are linear combination of real genes on a microarray dataset. Top ranked "virtual genes" are used for further analysis, e.g., sample classification. Our experiments with three public available datasets suggest that correlations between genes are indeed very good predictors of sample class labels. Unlike typical feature extraction algorithms, the "virtual gene" bears biological meaning: the weighted summation or difference of expression levels of several genes.

The rest of this paper is organized as follows. We present our "virtual gene" algorithm in Sec. 2. Both a synthetic and a real example from Alon dataset [1] are given. In Sec. 3, extensive experimental results are reported using three public available datasets. We give our conclusion and future work of this paper in Sec. 4.

2 Virtual Gene: A Gene Selection Algorithm

2.1 Gene Selection for Microarray Experiments

In this section we formalize the problem of gene selection for microarray datasets. Symbols used in this section will be used throughout this paper.

Let $\mathcal{G}$ be the set of all genes that are used in one study, $\mathcal{S}$ be the set of all experiments performed, $\mathcal{L}$ be the set of sample class labels of interest. We assume $\mathcal{G}, \mathcal{S}, \mathcal{L}$ are fixed for any given study. Let $n = |\mathcal{G}|$ be the total number of genes, $m = |\mathcal{S}|$ be the total number of experiments and $l = |\mathcal{L}|$ be the total number of class labels. Gene expression dataset used in our study can be defined as $\mathcal{E} = (\mathcal{G}, \mathcal{S}, L, E)$, where L is a list of sample class labels such that for $s \in \mathcal{S}$, $L(s) \in \mathcal{L}$ is the class label for sample s; expression matrix E is an $n \times m$ matrix of real numbers. $E(g,s)$, where $g \in \mathcal{G}, s \in \mathcal{S}$, is the expression level of gene g in experiment s. For simplicity of presentation, we use a subscripting scheme to refer to elements in $\mathcal{E}$. Let $\mathcal{E}(G,S) = (G,S,L',E')$ where $G \subseteq \mathcal{G}$ and $S \subseteq \mathcal{S}$. L' is a sublist of L containing class labels for samples S, E' is the subarray of E containing values of expression levels for genes G and experiments S. We also write $E' = E(G,S)$. We further use $L(S)$ to denote a list of class labels for the set of experiments S. Given training expression data $\mathcal{E}_{train} = (\mathcal{G}, S_{train}, L_{train}, E_{train})$, the problem of sample classification is to build a classifier that predict L_{new} for new experiment result $\mathcal{E}_{new} = (\mathcal{G}, S_{new}, L_{missing}, E_{new})$. $L_{missing}$ indicates that the class labels of samples S_{new} have not been decided yet. The problem of gene selection is to select a subset of genes $G' \subset \mathcal{G}$ based on $\mathcal{E}_{train}$ so that classifiers built from $\mathcal{E}_{train}(G', S_{train})$ predict L_{new} more accurately than classifiers built from $\mathcal{E}_{train}$. We use n' as the number of features being selected, or $n' = |G'|$.

2.2 An Example

Consider the following two examples as shown in Fig. 1. In each figure, the expression levels of two genes are monitored across several samples. Samples are labeled either cancerous or normal. In both cases, the expression levels of the selected genes vary randomly across the sample classes. However, their correlation is a good predictor of class labels. *Virtual gene* expression level is obtained using the Def. 2. In the case of Alon [1] dataset, the expression levels of H09719 are generally higher than that of L07648 in cancer tissues. In normal tissues, on the contrary, L07648 expresses consistently higher except in one sample. Such correlations could be good predictors of sample class labels. However, all feature selection algorithms listed in the previous section can not find and use such correlations. Single gene based algorithms will ignore both genes since neither of them is a good predictor of sample class labels in its own right. Correlation based algorithms will actually remove such correlations, should any of the genes have been selected.

2.3 Virtual Gene Algorithm

Definition 1. ***Virtual Gene*** *is a triplet $VG = (G_v, W, b)$ where $G_v \subseteq \mathcal{G}$ is a set of constituent genes, $|G_v| = n_v$, W is a matrix of size $n_v \times 1$, b is a numeric value. The expression levels of a* ***virtual gene*** *is determined using Definition 2.*

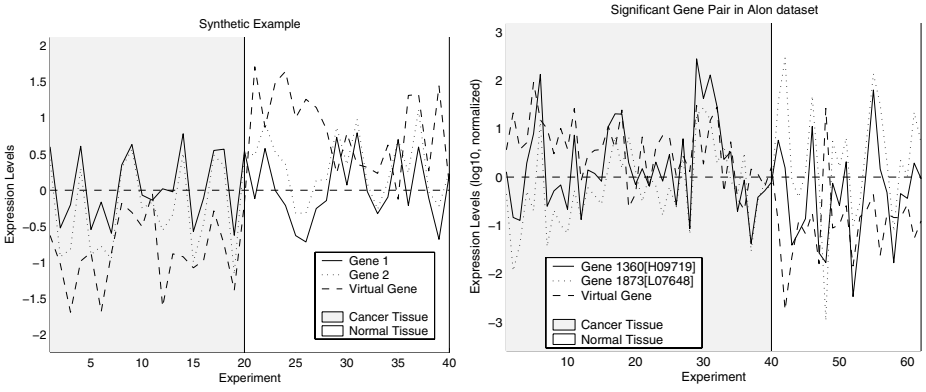


Fig. 1. Examples of gene pair being better predictor of class labels than single gene

Definition 2. ***(Virtual Gene Expression)*** *Given a* ***virtual gene*** $VG = (G_v, W, b)$ *and gene expression matrix E, where* $|G_v| = n_v$*, E is an* $n_v \times m_v$ *expression matrix, the* ***virtual gene expression*** VE *of a* virtual gene VG *is a linear combination of expression matrix E.* $VE(VG, E) = W' \times E + b$*, where* W' *is the transpose of* W*.*

A *virtual gene* is a triplet $VG = (G, W, b)$ defined as Def. 1. Parameters W and b are chosen using FLD (fisher linear discriminant) to maximize linear separability between sample classes as listed in Algorithm 1. Discriminative power of a *virtual gene expression* with respect to sample classes can be measured using normal single gene based scores. We use t-score in this paper for this purpose. *Pairwise virtual gene* is a special case of *virtual gene* where the number of genes involved is limited to two. In this case, only the correlations between a pair of genes are considered. By limiting *virtual gene* to gene pairs, computation can be carried out efficiently. According to our experiments, it performs well on three public available datasets.

Algorithm 1. *gen_vg* : Calculating Virtual Gene From Training Data

Require: $\mathcal{E} = (G, S, L, E)$ as gene expression data.
Ensure: $VG = (G, W, b)$ as a virtual gene.
1: $(W, b) \leftarrow fld(E, L)$, (W, b) is the model returned by fld algorithm.
2: **return** (G, W, b)

Definition 3. ***Pairwise virtual gene and its expression*** *are special cases for* ***virtual gene*** *and its expression, where the number of genes involved is limited to two.*

Exhaustive examination of all *pairwise virtual gene* requires $O(n^2)$ computation where n is the number of genes. For a large number of genes, exhaustive search of all gene pairs becomes inefficient. Such exhaustive search also invites unwanted noise since not all gene pairs bare biological meaning. For example, for genes that are expressed in different locations in a cell, in different biological processes, without biological interactions, their relative abundance may not be biologically significant. Ideally,

only gene pairs with some biological interaction shall be examined. We approximate this using a gene clustering approach. Each gene cluster corresponds roughly to some biological pathways. By limiting search among the gene pairs from the same gene cluster, we not only focus ourselves on these gene pairs that are more likely to interact biologically, but also make our gene selection algorithm much faster.

Algorithm 2. *pairwise_vg* : Pairwise Virtual Gene Selection

Require: $\mathcal{E} = (G, S, L, E)$; k as the number of genes to be selected; α; β
Ensure: VGS: as set of virtual genes $VG = (G, W, b)$

```
Initialize VGS to be an empty set. Initialize pair_score to be a sparse n × n array.
Cluster genes based on their expression levels in E. Result stores in Clusters.
for each gene cluster G' ∈ Clusters do
  for all gene g1 ∈ G' do
    for all gene g2 ∈ G' and g2 ≠ g1 do
      vg ← gen_vg(E((g1, g2), S))
      ve ← VE(vg, E((g1, g2), S))
      pair_score[g1, g2] ← t-score(ve, L)
    end for
  end for
end for
for i = 1 to k do
  (g1, g2) ← argmax_(g1,g2) (pair_score[g1, g2])
  vg ← gen_vg(E((g1, g2), S))
  add vg to VGS
  multiply pair_score that involves g1 or g2 by α.
  multiply pair_score that involves genes in same cluster of g1 or g2 by β.
  pair_score[g1, g2] ← minimum value
end for
return VGS
```

Algorithm 2 details *pairwise virtual gene selection* algorithm. Genes are first clustered based on their expression levels. For each pair of genes in the same cluster, virtual gene expression is calculated according to Def. 2. A single gene discriminative score with respect to the sample class labels is then derived from the virtual gene expression. All within-cluster pairwise virtual gene expression scores are calculated and stored for the next stage of analysis. The best scored virtual gene is then selected and pairwise scores are modified by two parameters. Pairwise scores of virtual genes that share constituent genes with the selected virtual gene are degraded by a constant α ranging $[0, 1]$. This dampens the effect of a single dominant salient gene. In the extreme case where α is set to 0, once a virtual gene is selected all virtual genes sharing constituent genes will not be further considered. The second parameter affecting the virtual gene selection is β, which controls how likely virtual genes in the same gene cluster are selected. Different gene clusters correspond to different regulative processes in a cell. Choosing genes from different gene clusters broadens the spectrum of the selected gene set. β also ranges $[0, 1]$. In the extreme situation where $\beta = 0$, only one virtual gene will be selected for each gene cluster. After modifying pairwise scores, the algorithm begins next

loop to find the highest scored virtual gene. This process repeats until k virtual genes have been selected. For performance comparison of *pairwise virtual gene* algorithm and single gene based algorithms, each *pairwise virtual gene* counts for two genes. For example, the performance of selecting 50 genes using single gene based algorithms would be compared to performance of selecting top 25 *pairwise virtual genes*.

2.4 Complexity of Pairwise Virtual Gene Algorithm

Pairwise virtual gene selection algorithm runs in three stages: (1) cluster genes based on expression profile (lines 1-2), (2) calculate discriminative scores for *pairwise virtual gene* (lines 3-11), and (3) select virtual genes with best discriminative scores (lines 12-20). We assume gene cluster number to be θ and n, m, k, α, β as discussed above.

In the first stage of analysis, k-means algorithm runs in $O(\theta n)$. In the second stage, the actual number of gene pairs examined is $O(\frac{n^2}{\theta})$, assuming gene clusters obtained in the previous stage are of roughly the same size. For each gene pair, the calculation of the virtual gene and its discriminative score require $O(m^2)$. Time complexity of the second stage is $O(\frac{m^2n^2}{\theta})$. Stage three requires $O(k(\frac{n^2}{\theta} + m^2 + n + \frac{n}{\theta}))$ time. Putting them together, we have time complexity of $O(\theta n + \frac{m^2n^2}{\theta} + k(m^2 + \frac{n^2}{\theta}))$. The most time consuming part in the previous expression is the term $O(\frac{m^2n^2}{\theta})$. In our experiments, we choose $\theta \sim \Theta(n)$. Considering the fact that $k < n$, the time complexity of Algorithm 2 becomes $O(n^2 + nm^2)$. The $O(n^2)$ term is for k-means clustering, which runs rather quickly. If no clustering is performed in stage 1 (or $\theta = 1$, one gene cluster), the time complexity becomes $O(n^2m^2 + kn^2)$. The savings in computation time is obvious.

Majority of space complexity for *virtual gene selection* algorithm comes from stage 2 in the algorithm where pairwise discriminative scores are recorded. The space needed for that is $O(\frac{n^2}{\theta})$ using sparse array. Under typical situation if we choose $\theta \sim \Theta(n)$, space complexity of Algorithm 2 becomes $O(n)$, although with a large constant.

3 Experiments

In order to assess the performance of our virtual gene feature selection algorithm, we use three benchmark datasets: colon cancer [1], leukemia [5] and multi-class cancer [12], all of which provide human gene expression levels using oligonucleotide microarrays. Samples are labeled with two class labels in the first two datasets. The multi-class dataset is labeled with multiple cancer types (and normal), of which we only use the cancer/normal distinction. Data are preprocessed using log transformation and normalization. Genes are filtered by fold change criteria. Performance of the feature selection method is measured by the classification accuracy of three different classifiers: KNN (k nearest neighbor), DLD (diagonal linear discriminant) and SVM (support vector machine, with linear kernel). Classification accuracy is estimated using cross validation process. We test four FSS algorithms, single gene t-score, single gene S2N score [5], pairwise t-score [3] and our virtual gene. For our algorithm, the number of gene clusters varies from 128 to 400 and α and β range from 0.8 to 1.

Table 1. Performance(%) of FSS methods using top 20 genes (10 virtual genes)

FSS	*Colon Cancer*			*Leukemia*		*Multi-class*	
method	*KNN*	*DLD*	*SVM*	*KNN*	*DLD*	*KNN*	*DLD*
t-score	81.58	80.10	81.35	94.97	92.75	78.92	78.34
S2N score	81.26	79.94	81.29	95.19	93.33	78.56	78.26
pairwise t-score	79.42	75.77	80.42	95.17	92.56	77.71	77.85
pairwise virtual gene	82.26	83.71	80.16	95.03	94.22	76.81	77.71

Table 2. Performance(%) of FSS methods using top 50 genes (25 virtual genes)

FSS	*Colon Cancer*			*Leukemia*		*Multi-class*	
method	*KNN*	*DLD*	*SVM*	*KNN*	*DLD*	*KNN*	*DLD*
t-score	81.52	78.10	81.81	95.47	94.14	81.81	78.85
S2N score	81.84	77.39	82.06	95.78	94.31	81.73	78.84
pairwise t-score	81.84	76.48	82.48	95.69	94.00	81.35	79.26
pairwise virtual gene	84.35	85.74	82.10	96.06	95.31	80.41	79.07

Results are summarized in the following tables. Table 1 and Table 2 show FSS performance when 20 or 50 genes (10 or 25 virtual genes) are selected for sample classification. Virtual gene algorithm performs comparatively well in all cases. In some cases, the virtual gene algorithm outperforms other algorithms by a large margin, indicating correlations of gene expression levels being good predictors of sample class labels.

4 Conclusion and Future Work

Gene selection is crucial both for building a good sample classifier and for selecting smaller gene set for further biological investigation. Feature extraction algorithms (PCA, SVD, etc.), single gene based discriminative scores (t-score, S2N, TNoM, information gain, etc.) and correlation based algorithms have been proposed for this purpose. In this paper, we proposed a totally different approach. Instead of trying to minimize correlations within the selected gene set, we examined whether such correlations are good predictors of sample class labels. *Virtual gene* is a linear combination of a set of real genes. Our experiments confirm our assumption that the correlations between genes are indeed good predictors of sample class labels, better in many cases than single gene based discriminative scores. There are biological explanation for this: genes interact with each other. The relative abundance of genes is a better predictor than the absolute values. Using gene clustering algorithm to limit gene pair selection seems promising. Our experiments show that by calculating pairwise scores for only a very small portion (0.5%) of all possible gene pairs, decent classification performance can still be achieved. This in turn shows most useful pairwise correlations are contained within gene clusters.

Our algorithm still has space for improvement. First but not least, we are interested in combining single gene based scores and virtual gene. In contrast to correlation based gene selection approaches, we can select top genes with high individual scores and top

correlations between genes. We also want to examine larger virtual genes, virtual genes that combine more than two genes. Gene clustering is only a crude way of grouping co-regulated genes. We are currently working on using gene ontology as a way to group genes. Our algorithm is quite open, several other algorithms (e.g., cluster analysis and discriminative power of single gene) can be plugged into our algorithm without much modification. We leave this as future work as well.

References

1. U. Alon, N. Barkai, et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissue probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. U.S.A.*, 96(12):6745–50, 1999.
2. A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. volume 7, pages 559–83, 2000.
3. T.H. Bø and I. Jonassen. New feature subset selection procedures for classification of expression profiles. *Genome Biology*, 3(4):research0017.1–0017.11, 2002.
4. G. V. Bobashev, S. Das, and A. Das. Experimental design for gene microarray experiments and differential expression analysis. *Methods of Microarray Data Analysis II*, 2001.
5. T. R. Golub et al. Molecular classifications of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–7, 1999.
6. T. Hastie, R. Tibshirani, et al. 'gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1(2), 2000.
7. j. Khan, J.S. Wei, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–9, 2001.
8. J. Jaeger, R. Sengupta, and W. L. Ruzzo. Improved gene selection for classification of microarrays. In *Proc. PSB*, 2003.
9. A .K. .Jian, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.
10. D. Jiang and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.
11. W. Li and I. Grosse. Gene selection criterion for discriminant microarray data analysis based on extreme value distributions. In *Proc. RECOMB*, 2003.
12. S. Ramaswamy, P. Tamayo, et al. Multiclass cancer diagnosis using tumor gene expression signatures. *PNAS*, 98(26):15149–15154, 2001.
13. Virginia Goss Tusher, Robert Tibshirani, and Gilbert Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98(9):5116–5121, April 2001.
14. Y. Wu and A. Zhang. Feature selection for classifying high-dimensional numerical data. In *CVPR 2004*, volume 2, pages 251–258, 2004.
15. E. P. Xing, M. I. Jordan, and R. M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proc. ICML 2001*, pages 601–608, 2001.
16. L. Yu and H. Liu. Redundancy based feature selection for microarray data. In *Proc. of SIGKDD*, 2004.

Bulk Synchronous Parallel ML: Modular Implementation and Performance Prediction

Frédéric Loulergue, Frédéric Gava, and David Billiet

Laboratory of Algorithms, Complexity and Logic – University Paris XII
61, avenue du Général de Gaulle – 94010 Créteil cedex – France
http://bsmllib.free.fr

Abstract. BSML is a library for parallel programming with the functional language Objective Caml. It is based on an extension of the λ-calculus by parallel operations on a parallel data structure named parallel vector. The execution time can be estimated, dead-locks and indeterminism are avoided. Programs are written as usual functional programs (in Objective Caml) but using a small set of additional functions. Provided functions are used to access the parameters of the parallel machine and to create and operate on parallel vectors. It follows the execution and cost model of the Bulk Synchronous Parallel model. The paper presents the lastest implementation of this library and experiments of performance prediction.

1 Introduction

The design of parallel programs and parallel programming languages is a trade-off. On one hand the programs should be efficient. But the efficiency should not come at the price of non portability and unpredictability of performances. The portability of code is needed to allow code reuse on a wide variety of architectures and to allow the existence of legacy code. The predictability of performances is needed to guarantee that the efficiency will always be achieved on any architecture.

Another very important characteristic of parallel programs is the complexity of their semantics. Deadlocks and indeterminism often hinder the practical use of parallelism by a large number of users. To avoid these undesirable properties, a trade-off has to be made between the expressiveness of the language and its structure which could decrease the expressiveness.

Bulk Synchronous Parallelism [15, 1] (BSP) is a model of computation which offers a high degree of abstraction like PRAM models but yet a realistic cost model based on a structured parallelism: deadlocks are avoided and indeterminism is limited to very specific cases in the BSPlib library [8]. BSP programs are portable across many parallel architectures.

Our research aims at combining the BSP model with functional programming. We obtained the Bulk Synchronous Parallel ML language (BSML) based on a *confluent* extension of the λ-calculus. Thus BSML is deadlock free and deterministic. Being a high-level language, programs are easier to write, to reuse and to

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1046–1054, 2005.
© Springer-Verlag Berlin Heidelberg 2005

compose. It is even possible to *certify* the correctness of BSML programs [5] with the help of the Coq proof assistant. The performance prediction of BSML programs is possible. BSML has been extended in many ways throughout the years and the papers related to this research are available at `http://bsml.free.fr`.

In section 2 we present the core of BSML, which is in fact currently a library, the BSML library, for the Objective Caml language [12]. Section 3 gives an overview of the new modular implementation of the current BSML library. Performance prediction is considered in section 4: we first describ how the BSP parameters of a parallel machine could be benchmarked and then we present a small experiment. Related work is presented in section 5. We conclude in section 6.

2 The BSML Library

There is currently no implementation of a full BSML language but rather a partial implementation as a library for Objective Caml language [12]. We assume the reader has basic knowledge about functional programming with Objective Caml and about the Bulk Synchronous Parallel model [1].

2.1 Primitives

BSML does not rely on SPMD programming. Programs are usual "sequential" Objective Caml programs but work on a parallel data structure. Some of the advantages is a simpler semantics and a better readability: the execution order follows (or at least the results is such as the execution order seems to follow) the reading order.

The core of the BSML library is based on the following elements:

bsp_p: unit→int
mkpar: (int →α) →α **par**
apply: (α →β) **par** →α **par** →β **par**
type α option = None | Some **of** α
put: (int→α option) **par** →(int→α option) **par**
at: α **par** →int →α

It gives access to the BSP parameters of the underling architecture. In particular, **bsp_p**() is p, the *static* number of processes. There is an abstract polymorphic type α **par** which represents the type of p-wide parallel vectors of objects of type α one per process. The nesting of **par** types is prohibited. Our type system enforces this restriction [6].

The BSML parallel constructs operates on parallel vectors. Those parallel vectors are created by **mkpar** so that (**mkpar** f) stores (f i) on process i for i between 0 and $(p-1)$. We usually write f as **fun** pid→e to show that the expression e may be different on each processor. This expression e is said to be *local*. The expression (**mkpar** f) is a parallel object and it is said to be *global*.

A BSP algorithm is expressed as a combination of asynchronous local computations and phases of global communication with global synchronization. Asynchronous phases are programmed with **mkpar** and **apply**.

The expression (**apply** (**mkpar** f) (**mkpar** e)) stores ((f i)(e i)) on process i.

Let consider the following expression:

```
let vf = mkpar(fun i→(+) i) and vv = mkpar(fun i→2*i+1) in
apply vf vv
```

The two parallel vectors are respectively equivalent to:

fun x→x+0	fun x→x+1	···	fun x→x+(p−1)

and

0	3	···	$2 \times (p-1)+1$

The expression **apply** vf vv is then evaluated to:

0	4	···	$2 \times (p-1)+2$

Readers familiar with BSPlib [8] will observe that we ignore the distinction between a communication request and its realization at the barrier. The communication and synchronization phases are expressed by **put**. Consider the expression:

$$\textbf{put}(\textbf{mkpar}(\textbf{fun}\ \mathsf{i}{\rightarrow}\mathsf{fs}_i)) \tag{1}$$

To send a value v from process j to process i, the function fs_j at process j must be such that (fs_j i) evaluates to Some v. To send no value from process j to process i, (fs_j i) must evaluate to None. Expression (1) evaluates to a parallel vector containing a function fd_i of delivered messages on every process. At process i, (fd_i j) evaluates to None if process j sent no message to process i or evaluates to Some v if process j sent the value v to the process i.

The synchronous projection operation **at** is such as (**at** vec n) return the n^{th} value of the parallel vector vec. **at** expresses communication and synchronization phases. The projection should not be evaluated inside the scope of a **mkpar**. This is enforced by our type system [6].

For the pure functional subset of Objective Caml, BSML is also purely functional: there are *no side-effects*. Moreover these four parallel functions are called *primitives* because they need lower-level libraries (C libraries wrapped to Objective Caml code) to be implemented. In the modular version of BSML the implementation of these functions rely either on MPI [14], PUB [2], PVM [7] or on the TCP/IP functions provided by the Unix module of Objective Caml.

2.2 Examples

Other functions are very often used when one write BSML programs. Nevertheless these functions can be defined using *only* the primitives and require no lower-level libraries. They are part of what is called the standard library of BSML.

For example the function replicate creates a parallel vector which contains the same value everywhere.

```
let replicate x = mkpar(fun pid→x)
```

It is also very common to apply the same sequential function at each process. It can be done using the parfun functions: they differ only in the number of arguments of the function to apply:

```
let parfun f v = apply (replicate f) v
let parfun2 f v1 v2 = apply (parfun f v1) v2
```

The semantics of the total exchange function is given by:

$$\text{totex} \langle\, v_0 \,, \ldots,\, v_{p-1} \,\rangle = \langle\, f \,, \ldots,\, f \,, \ldots,\, f \,\rangle$$

where f is such as $\forall i.(0 \leq i < p) \Rightarrow (f\ i) = v_i$. The code is as follows where compose is usual composition of functions:

```
(* totex: α par →(int →α ) par *)
let totex vv = parfun(compose noSome)(put(parfun (fun v dst→Some v) vv))
```

Its parallel cost is $(p-1) \times s \times g + L$, where s denotes the size of the biggest value v held at some process n in words.

From it we can obtain a version which returns a parallel vector of lists:

```
(* totex_list: α par→α  list par *)
let totex_list v = parfun2 List.map (totex v) (replicate(procs()))
```

where List.map $f\ [v_0; \ldots; v_n] = [(f\ v_0); \ldots; (f\ v_n)]$ and procs() $= [0; \ldots; p-1]$.

In section 4 we will use the following parallel reduction function:

```
let fold_direct f vec = let fold h::t=List.fold_left f h t in
parfun fold (totex_list vec)
```

where List.fold_left is a sequential reduction on lists in the Objective Caml standard library. Its BSP cost formula is (assuming op has a small and constant cost): $2 \times p - 1 + (p-1) \times s \times g + L$.

3 An Overview of the Implementation

In the implementation of the BSML library version 0.3, the Bsmllib module which contains the primitives presented in section 2.1, is implemented in SPMD style using a lower level communication library. This module called Comm is based on the following main elements:

```
pid: unit →int    nprocs: unit →int    send: α option array →α option array
```

There are several implementations of Comm based on MPI [14], PVM [7], BSPlib [8], PUB [2], TCP/IP, etc. The implementation of all the other modules of the BSML library, including the core module, is independent on the actual implementation of Comm, but only depends on its interface.

The meaning of pid and nprocs is obvious, they give respectively the process identifier and the number of processes of the parallel machine. send takes on each process an array of size nprocs(). These arrays contain optional values. The input values are obtained by applying at each process i, the function f_i (argument of the **put** primitive) to integers from 0 to (nprocs()−1).

If at process j the value contained at index i is (Some v) then the value v will be send from process j to process i. If the value is the None value, nothing will be sent. In the result, which are also arrays, None at index j at process i means than the process j sent no value ot i and a non (Some v) value means that process j sent the value v to i. A global synchronization occurs inside this communication function. **put** and **at** are implemented using send.

The implementation of the abstract type, **mkpar** and **apply** is as follows:

type α **par** $= \alpha$ **let mkpar** f = f (pid()) **let apply** f v = f v

For the MPI version, the pid function does not call the MPI_Comm_rank function each time. The MPI function is called only one time at initialization and the value is store in a reference. The pid function only obtains the value of this reference.

Let see how the **put** primitive works. Consider a parallel machine with 4 processors and functions f_i whose types are int$\rightarrow\alpha$ **par** such as $(f_i\ (i+1)) =$ Some v_i for $i = 0, 1, 2$ and $(f_i\ j) =$ None otherwise.

The expression **mkpar(fun** i $\rightarrow f_i$) would be evaluated as follows:

1. First at each process the function is applied to all process identifiers to produce p values, the messages to be sent by the processes. In the following figure, on the left side, a column represents the values produced at one process and the lines are ordered by destination (first line represents the messages to be sent to process 0, etc.).

2. Then the exchange of messages is actually performed. If we think of the table as a matrix, the resulting matrix is obtained by transposition (right side):

None	None	None	None
Some v_0	None	None	None
None	Some v_1	None	None
None	None	Some v_2	None

None	Some v_0	None	None
None	None	Some v_1	None
None	None	None	Some v_2
None	None	None	None

This operation is performed by the send function of the Comm module.

3. Finally the parallel vector of functions is produced. Each process i holds an array a_i of size p (a column of the previous matrix) and the function is **fun** x $\rightarrow a_i$.(x). In our example at process 3, $(f_3\ 0) =$ None which means that process 3 received no message from process 0 and $(f_3\ 2) =$ Some v_2 which means that process 2 sent the value v_2 to process 3.

4 Performance Prediction

One of the main advantages of the BSP model is its cost model: it is rather simple but yet accurate. Several papers (for example [10]) have demonstrated this fact using the BSPlib [8] library or the Paderborn University BSP Library (PUB) [2]. In his book [1], Bisseling presents in the first chapter the BSP model, programming using the BSPlib library and examples. Among this examples, the "probe" program is a benchmark used to determine the parameters of the BSP machines.

4.1 Benchmarking the BSP Parameters

There are four BSP parameters. Of course, the number of processors does not need to be benchmarked. The g and L parameters are expressed as multiples of the processors speed r. This parameter is the first to be determined. It is also the most difficult to determine accurately.

In [1], the target applications belong to the scientific computing area. Thus r is obtained by timing several computations using floating point operations. The first version of our bsmlprobe program does almost the same, because we would like to perform some comparisons with the C version (we use an average of the measured speeds, called r and the best of the speeds, called r'). But of course, arrays are not the most commonly used data structure in Caml. For applications with rich data structures, this value of r may be not very accurate. This way to obtain r is also more favorable to C as more complex data structures, or even function calls are more efficient in Objective Caml than in C. Thus in practice, for a general usage, the way to determine r should not only rely on floating arithmetics.

Then g and L are determined as follows: several super-steps are timed, with an increasing h relation. In these super-steps, each process sends to each other process n/p or $(n/p)+1$ words. Then the least squares method is used to obtain g and L.

The average results obtained by running the probe (C+MPI) and bsmlprobe (BSML 0.3 with MPI implementation of the Comm module) programs 10 times on a 6 Pentium IV nodes cluster interconnected with a Gigabit Ethernet network are as follows:

- **C+MPI:** $r = 478$ Mflops, $g = 25.2$ Flops/word, $L = 623141$ Flops
- **BSML(MPI):** $r = 469$ Mflops, $g = 28$ Flops/word, $L = 227512$ Flops

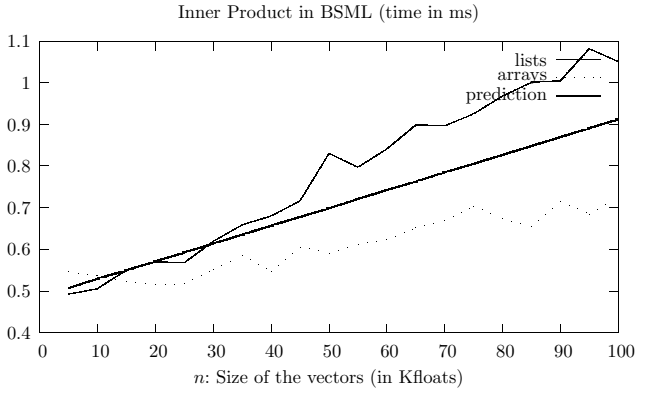
4.2 Experiments

We then performed some experiments on the program which compute the inner product of two vectors. There are two versions, one using arrays to store the vectors and one using lists to store them:

```
let inprod_array v1 v2 = let s = ref 0. in
    for i = 0 to (Array.length v1)−1 do s:=!s+.(v1.(i)∗.v2.(i)); done; !s
let inprod_list v1 v2 = List.fold_left2 (fun s x y→s+.x∗.y) 0. v1 v2
let inprod seqinprod v1 v2 = let local_inprod = parfun2 seqinprod v1 v2 in
  Bsmlcomm.fold_direct (+.) local_inprod
```

This code uses the fold_direct function from the BSML standard library (presented in section 2.2). Thus the overall BSP cost formula of inprod is: $n + 2 \times p + (p-1) \times g + L$.

These programs were run 15 times each for increasing size of vectors (from 5.000 to 100.000) and the average was taken. The following figure summarizes the timings. The predicted performances using the parameters obtained in the previous section are also given:



5 Related Work

The first libraries devoted to BSP programming were libraries for C which converged to a standard proposal and the Oxford BSPlib [8]. Being a library for C, it is of course very different from BSML. First the programming style is SPMD and the communication function bsp_put in *Direct Remote Memory Access* (DRMA) style may cause indeterminism if two processors try to write into the same memory zone of a third processor. The synchronization barrier should be explicitly called with bsp_sync. There are no updates of this library since 1998. The *Paderborn University BSPlib* (PUB) [2] library is also dedicated to BSP programming. It offers additional features not part of the standard. For example it is possible to have threads and the library takes care of avoiding the mismatch of messages.

There are other libraries for *Coarse-Grained Multicomputer* (CGM) programming. CGM is a special case of the BSP model and some of these libraries allow to program BSP algorithms which are not CGM algorithms [3]. NestStep [11] is an extension of C, C++ and Java for SPMD programming of BSP algorithms. It is based on a virtual shared memory on top of message passing libraries.

To our knowledge, two algorithmic skeletons languages are based on the BSP model [16, 4]. They have of course the advantages and drawbacks of skeleton approaches. Implementations have not been released. VEC-BSP has the advantage of having a cost algebra associated with the language which allows automatic performance prediction.

BSP-Haskell [13] and BSP Python [9] are inspired from the BSλ-calculus and BSML. BSP Python adapts BSML to Python which is an object oriented language. Thus elements of BSML are present. There are parallel vectors which are instances of a global class which has a method **put** for communications, *etc.*. The constructor of the ParData class takes as argument a function like BSML **mkpar**. BSP-Haskell relies on monads and it allows to avoid the evaluation of nested parallel vectors.

6 Conclusions

The BSML library allows declarative parallel programming in a safe environnement. Being implemented using Objective Caml and using a modular approach which allows to perform the communications with various communication libraries, it is portable, and efficient, on a wide range of architectures. The basic parallel operations of BSML are Bulk Synchronous Parallel operations, thus allow accurate and portable performance prediction.

Acknowledgements

This work is supported by the ACI Grid program from the French Ministry of Research, under the project CARAML (www.caraml.org).

References

1. R. Bisseling. *Parallel Scientific Computation. A structured approach using BSP and MPI.* Oxford University Press, 2004.
2. O. Bonorden, B. Juurlink, I. von Otte, and O. Rieping. The Paderborn University BSP (PUB) library. *Parallel Computing*, 29(2):187–207, 2003.
3. A. Chan and F. Dehne. CGMgraph/CGMlib: Implementing and Testing CGM Graph Algorithms on PC Clusters. In *Proceedings of the 10th EuroPVM/MPI conference*. Springer, 2003.
4. M. Cole and Y. Hayashi. Static Performance Prediction of Skeletal Programs. *Parallel Algorithms and Applications*, 17(1):59–84, 2002.
5. F. Gava. Formal Proofs of Functional BSP Programs. *Parallel Processing Letters*, 13(3):365–376, 2003.
6. F. Gava and F. Loulergue. A Static Analysis for Bulk Synchronous Parallel ML to Avoid Parallel Nesting. *Future Generation Computer Systems*, 2005. to appear.
7. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*. Scientific and Engineering Computation Series. MIT Press, 1994.
8. J.M.D. Hill, W.F. McColl, and al. BSPlib: The BSP Programming Library. *Parallel Computing*, 24:1947–1980, 1998.
9. K. Hinsen. High-Level Parallel Software Development with Python and BSP. *Parallel Processing Letters*, 13(3):461–472, 2003.
10. S.A. Jarvis, J.M.D Hill, C.J. Siniolakis, and V.P. Vasilev. Portable and architecture independent parallel performance tuning using BSP. *Parallel Computing*, 28:1587–1609, 2002.
11. Christoph Kessler. Managing distributed shared arrays in a bulk-synchronous parallel environment. *Concurrency and Computation: Practice and Experience*, 16:133–153, 2004.
12. X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon. The Objective Caml System release 3.08, 2004. web pages at www.ocaml.org.
13. Q. Miller. BSP in a Lazy Functional Context. In *Trends in Functional Programming*, volume 3. Intellect Books, may 2002.

14. M. Snir and W. Gropp. *MPI the Complete Reference*. MIT Press, 1998.
15. Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, August 1990.
16. A. Zavanella. Skeletons, BSP and performance portability. *Parallel Processing Letters*, 11(4):393–405, 2001.

Fast Expression Templates

Object-Oriented High Performance Computing

Jochen Härdtlein, Alexander Linke, and Christoph Pflaum

University of Erlangen, Department of Computer Science 10,
System Simulation Group, Cauerstr. 6, D-91058 Erlangen, Germany

Abstract. Expression templates (ET) can significantly reduce the implementation effort of mathematical software. For some compilers, especially for those of supercomputers, however, it can be observed that classical ET implementations do not deliver the expected performance. This is because aliasing of pointers in combination with the complicated ET constructs becomes much more difficult. Therefore, we introduced the concept of enumerated variables, which are provided with an additional integer template parameter. Based on this new implementation of ET we obtain a C++ code whose performance is very close to the handcrafted C code. The performance results of these so-called *Fast ET* are presented for the Hitachi SR8000 supercomputer and the NEC SX6, both with automatic vectorization and parallelization. Additionally we studied the combination of *Fast ET* and OpenMP on a high performance Opteron cluster.

1 Introduction

While object-oriented programming is being embraced in industry, its acceptance by the High Performing Computing Community is still very low, mainly because of supposed performance losses. Indeed, introducing abstract data types and operator overloading empowers the software engineer to forge interfaces very close to problem-specific notation, which can be understood by C++-compilers. However, there are no language features in C++ which are designed to inform the compiler about allowed transformations of mathematical expressions involving user-defined abstract data types. Therefore, though such user-defined mathematical expressions can be compiled by C++-compilers, they often perform very poorly. The first solution to overcome these performance problems was represented by the use of ET in C++, whose development we will summarize in the following.

Veldhuizen and Vandervoorde concurrently recognized the potential of templates in C++ and, in 1995 and 1996, Veldhuizen published his first articles about template meta programming [10] and ET [11]. ET were proposed to overcome performance problems which arise from simple operator overloading in mathematical expressions. Unnecessary temporaries are avoided by performing some kind of expression-dependent inlining within a single loop. Therefore, less data

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1055–1063, 2005.
© Springer-Verlag Berlin Heidelberg 2005

is pumped through the memory hierarchy during the evaluation, thus enhancing code performance. For many systems the efficiency of ET implementations competes with their Fortran counterparts [12].

Soon, there was a rapid development of powerful mathematical packages based on ET; e.g., Blitz++ by Veldhuizen [13] and the Generative Matrix Computation Library (GMCL) described in [3]. Further important software frameworks are PETE [7] and POOMA [9], which represent two projects started at the Los Alamos National Laboratory. PETE is a tool for implementing ET for various applications. POOMA supports the implementation of mathematical algorithms for solving partial differential equations. Users of POOMA write sequential source code similar to FORTRAN 90 and benefit from getting automatic parallelization on various platforms just by using compiler switches. Some aims of the POOMA project are shared by the EXPDE project [8]. EXPDE eases the implementation of parallel 3D finite element codes by providing finite element and multigrid operators on arbitrarily shaped domains. Mathematical algorithms can be formulated with EXPDE in a language very close to the mathematical language. Furthermore, ET enable automatic parallelization.

Performance problems with ET were discovered for the first time by Bassetti, Davis, and Quinlan in 1997, see [1] and [2]. They studied the register spillage of ET codes and figured out the downsides of these implementations. Additionally to that paper one can observe thoroughgoing problems using ET on computers that enable automatic parallelization; e.g., vector machines or multiprocessor machines using OpenMP. Analyzing these problems, we developed, via the concept of enumerated variables, an implementation technique to assist the compiler overcoming these problems.

In the next section we will demonstrate how classical ET work and show the problems from which they suffer. After that we will present our implementation of Fast ET. In section 4 we will provide some examples comparing the described implementations of ET with C code computing the same problems. Section 5 concludes the paper.

2 Classical Expression Templates

For the sake of simplicity, we will focus on a minimal classical ET implementation for vectors with component-wise multiplication, as suggested in [14]. The mathematical operator `Times` is encapsulated by:

```
struct Times{
  static inline double apply(double a, double b) {return a*b;}
};
```

The expression class is then implemented as follows:

```
template<typename Left, typename Op, typename Right>
struct Expr {
  const Left &left_;  const Right &right_;
  Expr(const Left &t1, const Right &t2):left_(t1),right_(t2){}
```

```
  double Give(int i) const {
    return Op::apply(left_.Give(i), right_.Give(i));}
};
```

Here is a simple vector class:

```
class Vector {
  double * data_;
  const int N_;
 public:
  Vector(int N, double val);
  double Give(int i) const { return data_[i]; }

  template<typename Left, typename Op, typename Right>
  void operator= (const Expr<Left, Op, Right> &expr){
    for(int i=0; i < N_; ++i)
       data_[i] = expr.Give(i);
  }
};
```

Finally, the `operator*` is implemented as follows. The idea is to create a type that stores the information for the evaluation.

```
template<typename Left>
Expr<Left,Times, Vector>
 operator*(const Left &a,const Vector &b){
   return Expr<Left,Times, Vector>(a, b);
 }
```

We will now explain how ET work. Consider the line `c = a*b*a`. The type inference mechanism of the compiler can be represented as:

```
c = a*b*a;
  = Expr<Vector,Times,Vector>(a,b)*a;
  = Expr<Expr<Vector,Times,Vector>,Times,Vector>(
                          Expr<Vector,Times,Vector>(a,b),a);
  =: expr;
```

The call of `c.operator=` looks as follows:

```
c.operator=<Expr<Expr<Vector,Times,Vector>,Times,Vector> >(expr){
          for(int i=0; i < N_; ++i) data_[i] = expr.Give(i);}
```

Now `expr.Give(i)` can be expanded by inlining `Give(i)` from each node of the expression tree:

```
data_[i] = expr.Give(i);
  = Times::apply(expr.left_.Give(i),expr.right_.Give(i));
  = Times::apply(Times::apply(expr.left_.left_.Give(i),
                 expr.left_.right_.Give(i)),expr.right_.Give(i));
  = Times::apply(Times::apply(expr.left_.left_.data_[i],
                expr.left_.right_.data_[i]),expr.right_.data_[i]);
  = expr.left_.left_.data_[i] * expr.left_.right_.data_[i]
                                          * expr.right_.data_[i];
```

If `operator=` is inlined and the aliasing for the pointers works, a C++ compiler can, in principle, optimize the expression such that it correspond to the following code fragment:

```
for(int i=0; i < N; ++i)
  c.data_[i] = a.data_[i] * b.data_[i] * a.data_[i];
```

This is just the code a C programmer would naturally write. Indeed, to reach this stage a compiler must have good optimization facilities. Since not all compilers are capable of C++-specific optimization facilities, the resulting programs based on classical ET can suffer from a variety of performance problems. Although component-wise multiplication of three vectors is a very trivial application of ET, the intermediate C++ code inlined by ET is quite complex. More difficult applications such as ET for discretized differential operators in 3D yield much more complex intermediate code.

The main problems of ET are discussed in [1] and [2]. For a better understanding of the performance lacks we focus again on the example presented above. The pointers `expr.left_.left_.data_` and `expr.right_.data_` represent the same data array. The aliasing concept of the C++ compiler must be able to resolve this to the pointers they denote. Otherwise there will be two different pointers that represent the same data. During the evaluation loads are issued, even if the data has already been loaded into the CPU core. This obviously causes performance lacks. Moreover, as aliasing is even more important on vector machines the solution of this problem can significantly reduce the performance problems in object-oriented high performance computing.

3 Fast Expression Templates

After this short discussion of classical ET and their problems, we present an alternative approach based the concept of enumerated variables. The idea is to equip the `Vector` class with an additional integer parameter which is unique for each vector variable. The aim is to communicate to the compiler if the variables differ or not. Our second idea is to derive an expression object from an expression whose information is completely accessible at compile time. This allows more intelligent inlining and guarantees enhanced C++ performance.

```
template <class A, class B>
struct Times : public Expr<Times<A,B> > {
  static inline double Give(int i){return A::Give(i)*B::Give(i);}
};
template <class A, class B>
inline Times<A,B> operator*(const Expr<A>& a, const Expr<B>& b){
  return Times<A,B>();
}
```

Moreover, the `Give` function for the `Vector` class has to be static as well. However, the vector components and even the vector size is still unknown at compile time. We solve this problem by introducing a static data pointer. This is

reasonable because different variables have different types. Consider the following implementation of the `Vector` class:

```
template<int num>
class Vector : public Expr<Vector<num> >{
  static double* data_;
  const int N_;
 public:
  ...
  template<class A>
  inline void operator=(const Expr<A>& a) const {
    for(int i = 0; i < N_; ++i) data_[i] = A::Give(i);
  }
  static inline double Give(int i) {return data_[i];}
};
template<int num> double* Vector<num>::data_;
```

Last but not least, the excerpt from the main program looks as follows:

```
Vector<0> a(N, 1.);  Vector<1> b(N, 3.);  Vector<2> c(N, 0.);
c = a*b*a;
```

We will now explain how this new approach differs from classical ET. To this end, we focus again on the expression `c=a*b*a`, whose expression object has the following type:

```
Times<Times<Vector<0>,Vector<1>>,Vector<0>>
```

Since this expression type contains the complete information about the expression `c=a*b*a`, more efficient inlining is possible. The expression is evaluated as:

```
c.operator=
  Times<Times<Vector<0>,Vector<1>>,Vector<0>>(expr){
   for(int i = 0; i < N; ++i)
     c.data_[i] = Times<Times<Vector<0>,Vector<1>>,Vector<0>>::Give(i);
}
```

Now we demonstrate how the static `Give(i)` function can be inlined:

```
Vector<2>::data_[i] =
  = Times<Times<Vector<0>,Vector<1>>,Vector<0>>::Give(i);
  = Times<Vector<0>,Vector<1>>::Give(i) * Vector<0>::Give(i);
  = Vector<0>::Give(i) * Vector<1>::Give(i) * Vector<0>::Give(i);
  = Vector<0>::data_[i]*Vector<1>::data_[i]*Vector<0>::data_[i];
```

This inlined source code corresponds to the code a programmer would naively write. In comparison to classical ET where the pointers `expr.right_.data_` and `expr.left_.left_.data_` represent the same data array, here the pointer `Vector<0>::data_` occurs twice. Hence, Fast ET support the compiler in aliasing identical data arrays.

Working only with the type of an expression means using only that information which is really needed to build the evaluation. Enumerating is just the base

for reaching this goal. In general the enumeration concept with static data joins two ideas of programming. On one hand, it introduces the use of global variables that are easier to optimize for compilers, because there are less aliasing conflicts than in pure object-oriented codes. On the other hand, enumerated variables enable data encapsulation in the object-oriented sense. And joining all in Fast ET we apply well performing operator overloading.

4 Performance Results

After the discussions of classical ET (CET) and Fast ET (FET) we present some performance results. The two implementations are compared with a C code that uses no ET (NET) but computes the same problem. In addition to the multiplication we implemented component-wise sum of vectors in the same manner. As we are interested in the effort of ET in high performance computing we tested these implementations to this aspects. We analyzed two types of expressions, at first the vector triad $a = b + c * d$ that does not suffer from the aliasing problem because each vector only occurs once. The second expression is $a = \sum_{i=1}^{7} a^i$. Obviously the efficiency of its implementation is highly dependent on the compiler's capabilities to detect array aliasing.

The results are presented for three platforms. At first the NEC SX-6/48M6 at the HLRS in Stuttgart, which is a shared memory vector system. We started the computation on one node with automatic vectorization. The sxc++ compiler enables the vectorization. The second platform is the Hitachi SR8000 supercomputer at the LRZ Munich. A single node is equipped with eight RISC processors and each processor can perform vector operations with floating point numbers, see [4]. On the Hitachi SR8000 we used the optimizing C++ compiler sCC by Hitachi, because it is the only C++ compiler which can vectorize C++ programs on this platform. At last we demonstrate the performance tests on a

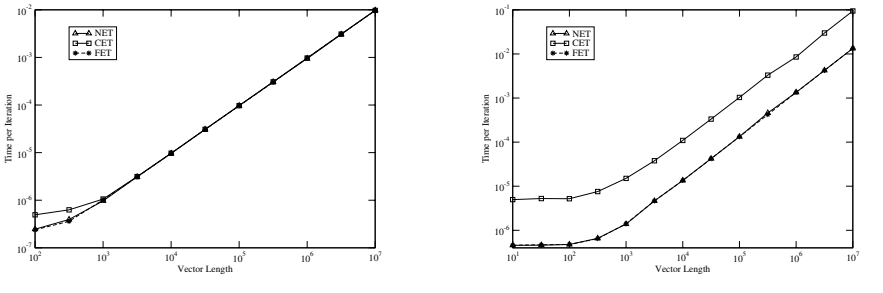


Fig. 1. Performance results on the Nec SX-6/48M6. On the left side the graph for the vector triad where only small vector lengths show the improvement by FET. The right figure presents the performance of the expression $a = \sum_{i=1}^{7} a^i$. FET run more than ten times faster than CET even for large vectors

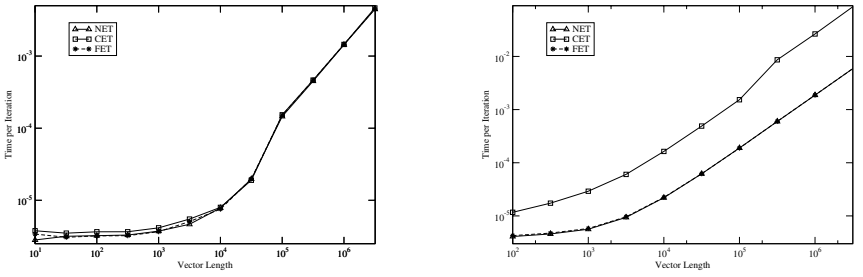


Fig. 2. $a = b + c * d$ and $a = \sum_{i=1}^{7} a^i$ on the Hitachi SR8000 supercomputer. The first example points out a small improvement by FET in comparison to CET mainly for small vector sizes. The right figure demonstrates the performance enhancement by the FET implementation

AMD Opteron Cluster with dual-nodes and quad-nodes. On this platform the implementations are translated by the Intel C++ compiler, version 8.1, and the evaluation is started on a dual node.

The results of the two examples are presented below. We measured the time per iteration over the vector length. At first, the vector triad does not suffer from lacks in aliasing. Only for small vector sizes the FET work faster than CET. This is reasonable because, unlike CET, the FET implementations do not store the expression members in the operating classes. However, this effect is compiler-dependent; e.g., the Intel C++ compiler optimizes all the nested function calls through the classical ET constructs in the expected way (see Fig. 3). Hence, there are also no differences in the performance for small vector lengths. The second example shows the real improvements by FET. In the majority of cases, they

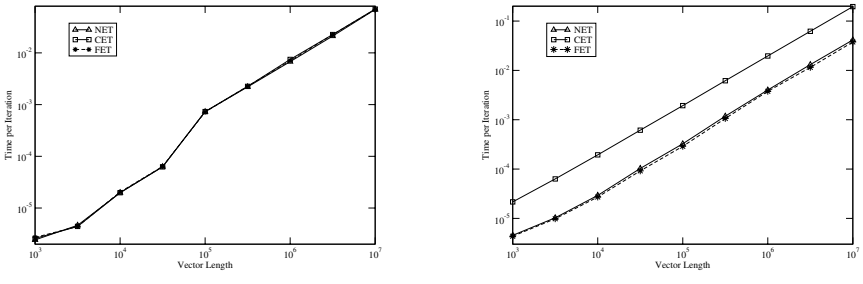


Fig. 3. The Opteron Cluster using Open MP and the Intel C++ compiler does not arise performance differences for the vector triad. This is due to the good optimizing facilities of the Intel compiler. However, the big expression in the right graph confirms the improvements resulting from the application of FET

work as well as the C code does. Only some small oscillations can be observed. The CET implementation suffers from the lacks in aliasing. The compiler does not recognize the repeated occurrence of the data pointer of the vector a. Hence, during the evaluation loads are issued even though the data has already been loaded into the CPU core. already in cache. This causes the performance lacks that slow down the implementation more than ten times compared to the other ones.

5 Conclusions and Perspectives

While ET have been in use for about eight years now, their problems discussed by Bassetti, Davis, and Quinlan led to a break in the euphoria. Recognizing the lacks of classical implementations, we introduced a concept to avoid these problems. Our new implementation of ET supports the compiler in solving the aliasing problems. Of course, there is an extra effort in organizing FET. Every template integer of an enumerated variable has to be unique within the whole program code, because they work, in principle, like global variables. Using FET in a library yields the enumeration of the variables to the user. A simple solution is the use of the `__LINE__` macro and defining every vector in a different line.

The performance that could be reached by using FET instead of the classical implementations is near to the C code performance. Hence, FET potentiates the ideas of a user-friendly interface and a well performing code.

References

1. Bassetti, F, Davis, K, and Quinlan, D: Towards Fortran 77 Performance from Object-Oriented C++ Scientific Framework: HPC '98 April 5-9, 1998.
2. Bassetti, F, Davis, K, and Quinlan, D: C++ Expression Templates Performance Issues in Scientific Computing. CRPC-TR97705-S, October 1997.
3. Czarnecki, K, and Eisenecker, U: Generative Programming : Methods, Tools, and Applications. Addison-Wesley, Boston, 2000.
4. Leibniz-Rechenzentrum München: The Hitachi SR8000-F1, System Description. http://www.lrz-muenchen.de/services/compute/hlrb/system-en
5. High Performance Computing Center Stuttgart: The NEC SX-6 Cluster Documentation. http://www.hlrs.de/hw-access/platforms/sx6/user_doc
6. Department of Computer Science 10, System Simulation, Erlangen: HPC Cluster http://www10.informatik.uni-erlangen.de/Cluster/hpc.shtml
7. Los Alamos National Laboratories: PETE - Portable Expression Templates Engine. http://www.acl.lanl.gov/pete/html/introduction.html
8. Pflaum, C: Expression Templates for Partial Differential Equations. Comput. Visual. Sci. **4**, 1–8, (2001).
9. Los Alamos National Laboratories: POOMA: www.acl.lanl.gov/pooma
10. Veldhuizen, T: Using C++ Template Metaprograms. *C++ Report* Vol. 7 No 4. (May 1995), pp. 36–43.
11. Veldhuizen, T: Expression Templates. C++ Report **7** (5), 26–31 (1995).

12. Veldhuizen, T: Will C++ be faster than Fortran? Proceedings of the 1st International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'97).
13. Veldhuizen, T: Blitz++. http://oonumerics.org/blitz/index.html
14. Veldhuizen, T: Techniques for Scientific C++. Indiana University Computer Science Technical Report No 542, Version 0.4, August 2000.

Solving Coupled Geoscience Problems on High Performance Computing Platforms

Dany Kemmler[1], Panagiotis Adamidis[2], Wenqing Wang[1], Sebastian Bauer[1], and Olaf Kolditz[1]

[1] Center for Applied Geoscience, University of Tuebingen, Germany
dani@uni-tuebingen.de
http://www.uni-tuebingen.de/zag/
[2] High Performance Computing Center, University of Stuttgart, Germany
http://www.hlrs.de

Abstract. This paper describes a finite element geo-process modeling software, which is able to solve multiphysics problems in the area of geo science. It is one of the first applications providing complete thermal, two-phase-flow, inelastic deformation court within one framework. As a real-life example the FEBEX TH^2M coupling problems are described. These kind of problems are very demanding in terms of CPU time and memory space which cannot be provided by single processor architecture. Therefore the exploitation of high performance architectures is required. Here we present the results of the vectorization of the existing serial code as a first step towards parallelization.

1 Introduction

Modeling complex coupled transient and dynamic processes found in geo-systems requires increasingly high resolution models for a more precise and qualified validation and evaluation. The more exactly the investigation of the given problem can be accomplished, the better the utilization, management and planning of the given geo-resources can be undertaken.

1.1 GeoSys/RockFlow

GeoSys/RockFlow (GS/RF) [1] is a finite element geo-process modeling software tool finding increased application in modeling and simulation in fields such as water resource management, geotechnics, design of geo-engineered barriers, exploitation of geothermal energy, soil and groundwater contaminant transport and remediation strategies. Its current design is based upon a sequential approach.

Here we present a new project which started in fall 2004 and is funded by the German Research Foundation (DFG) and the State of Baden-Wuerttemberg (BW). Its goal is to bundle the knowledge of the experts at the High Performance Computing Center (HLRS) in Stuttgart and the scientists and researchers working at the Center for Applied Geology, (ZAG) at the University of Tuebingen. This project is truly interdisciplinary and multinational as specialists in mathematics, computer science, hydrology, geo science, chemistry, just to name a few,

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1064–1071, 2005.
© Springer-Verlag Berlin Heidelberg 2005

are continuing to develop the already existing serial software and to finally transform it into a GRID application deploying massive-parallel systems in Stuttgart and Tuebingen.

2 GeoSys/RockFlow Model Features

The finite element simulator GS/RF covers a wide range of physical and chemical processes relevant to environmental hydrosystems. The processes can be grouped in 4 different categories as summarized below:

1. **Hydrological Processes:**
   - Groundwater flow in confined and unconfined aquifers
   - Multi phase flow
   - Fracture flow, dual porosity
   - Density dependent flow (thermal, tracer)
   - River flow (based on averaged 1-D Saint-Venant equations)
2. **Thermal Processes:**
   - Heat transport with density changes
   - Non isothermal multiphase flow with phase changes
3. **Chemical Processes:**
   - Multi-component transport with density changes
   - Sorption models
   - Reactive Transport (i.e. Freundlich Isotherm)
   - Chemical reactions via coupling to PHREEQC2
4. **Mechanical Processes:**
   - Poro elasticity
   - Thermo elasticity
   - Elasto plasticity (hardening)

Models can be created and run using a graphical user interface (Windows application). Built-in mesh generators are: gmsh, PrisGen and TetMesh. Meshing of complex structures can be done using gOcad or the pre-processor GINA developed by the Federal Institute for Geosciences and Natural Resources (BGR). Hybrid meshing is possible. ArcGIS shape files can be read and converted to polylines which are then used to create meshes or assign boundary or initial conditions etc. An interface to Gstat allows for the generation of three dimensional heterogeneous conductivity fields. Two dimensional contour plots and time-value graphs are displayed during the simulation in the GUI or Tecplot output files are written directly or created with the post-processor RF2TP also developed by the BGR.

Despite its long history dating back to 1985 the code is programmed according to recent programming principles. The software was constantly improved. In 2003/2004 the code underwent a major re-organization to benefit even more from object-orientation and to allow an easier switching between process couplings. Most recent changes are: use of C++, organization of RockFlow into GeoSys: GEOLib, MSHLib, FEMLib and the creation and encapsulation of process-oriented objects (PCS). These changes provide a solid basis for further program development within a growing research team.

3 The FEBEX Experiment

As an example of the necessity of high performance computing in environmental science we present results of different simulations of the FEBEX experiment [2] which is a full scale engineering barriers experiment in crystalline rock (Fig. 1) for high level radioactive waste (HLW) repository.

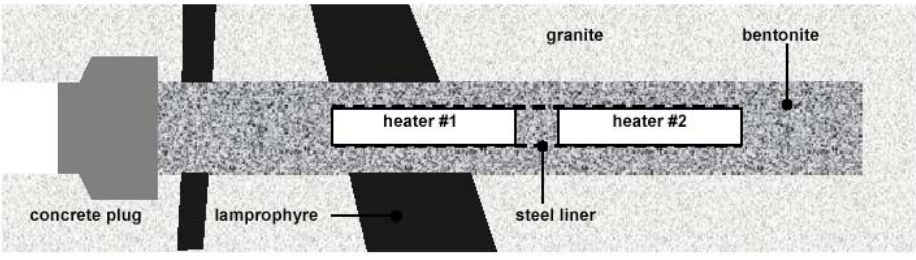


Fig. 1. Layout of the FEBEX experiment

In the experiment, the heaters experienced a temperature of 100°C. Bentonit was used as filling material (geotechnical barrier) around the heater. The heaters and geotechnical barrier were in a tunnel (2.4 m in diameter) surrounded by granite. The bentonite swells as a result of the intrusion of ground water from fractures. The swelling effect changed the microstructure of bentonite, i.e. porosity and in particular permeability of the bentonite. On the other hand it increased the swelling pressure and accordingly the effective tension in the bentonite. The expanded bentonite in turn exerted pressure on the surrounding granite and affected the stress conditions and possibly the water path in it. Gas was produced by the high temperature of the heaters from the water, which moved outward. The experiment was conducted in order to understand the thermo-hydro-mechanically (THM) coupled phenomena in the bentonite and surrounding granite and furthermore provide a valid reference for long term HLW repository.

3.1 Simulation Results

One part of the simulation was the analysis of the TH^2M coupling problems (always with the same boundary conditions and with the same material parameters) using a fine mesh of 3276 nodes and 2640 hexahedra. The geometry and mesh are depicted in Fig. 2, in which a cross section close to the front of the present saturation propagation is selected to portray the inside domain and z direction points downward in order to permit a close look around the canister surface.

In the following figures (Fig. 3) the primary variables of the TH^2M simulations after 10 ($t = 10^5$ sec) time steps are depicted.

The FEBEX simulations are to our knowledge the first all-in-one 3D fully coupled TH^2MC^n simulations in which the complete thermal, two-phase-flow,

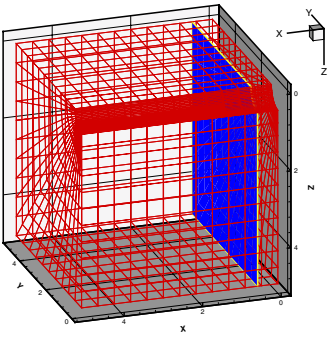


Fig. 2. Mesh and slice section

inelastic deformation court is provided by one unique code. TH^2MC^n simulations are very sensitive to time and space scale discretizations and such simulations are very expensive with respect to computation time.

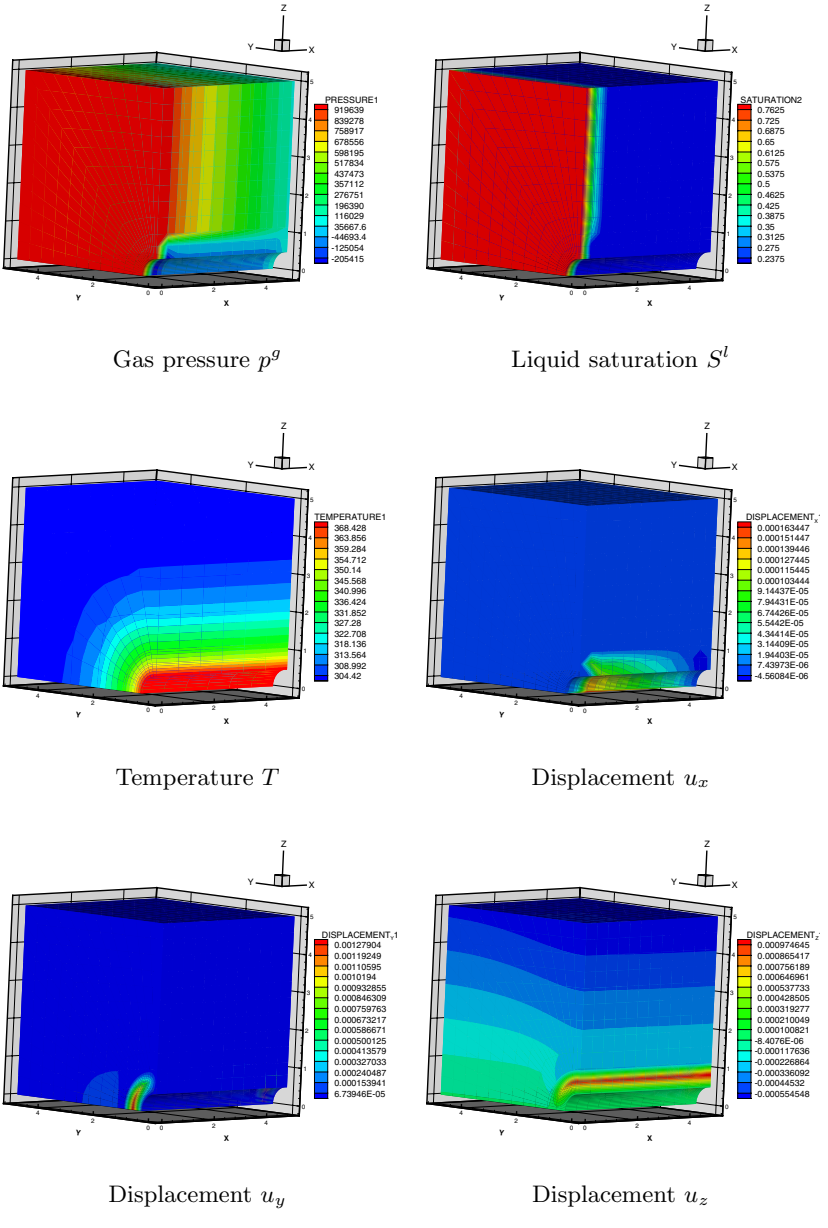
3.2 FEBEX Performance

The results obtained from this analysis show strong TH^{2}M coupling phenomena in the present FEBEX type problem. The change of displacements and stresses is large in the buffer. This change is partially due to the material properties of the buffer but mainly it is induced by stiff boundary condition around the buffer. Therefore, the safety assessment must be focused on this domain. This computation required much more CPU time due to the necessity of solving four algebraic equations within coupling iterations, nonlinear iterations for different processes. For example, a converged Newton step for plasticity deformation took an average of 2.5 hours using a machine with an Intel III 2.0GHz CPU. To compute 100 time steps of FEBEX on 4 processors the serial version of GS/RF runs nonstop for one week. To improve computation efficiency high performance computing on powerful parallel machines has to be deployed.

4 Parallelization Strategy

Today's modern single processor machines are not powerful enough to process large scale models necessary for the realistic simulation of in situ highly complex geoscience systems.

Through the combination of the wide field of geoscience and high performance computation supported through effective algorithms found in computer science a complete new level in simulation and modeling can be attained. The final goal is to compute models consisting of more than ten million of elements with a higher grade of specialization.



Gas pressure p^g | Liquid saturation S^l

Temperature T | Displacement u_x

Displacement u_y | Displacement u_z

Fig. 3. Primary variables of TH^{2}M simulations for $t = 10^5$sec

4.1 Computing Platform

The target parallel platform for the work presented in this paper is a GRID consisting of three parallel computing platforms, located at the computing center of

the University of Tuebingen, the HLRS in Stuttgart and a institute-own Linux cluster. As the most powerful environment within the GRID can be found in Stuttgart we have ported a first version of GS/RF on a NEC SX6 at the HLRS. The SX6, in summer being replaced by a SX8, implements a parallel architecture with distributed memory, in particular a cluster of symmetric multi-processors (SMPs). Each of its SMP nodes consists of 8 vector processors. To take advantage of the features of such an architecture the code has to be vectorised and parallelized.

In order to run an application on such a computational environment the work and the data have to be distributed among the processors. The decision that has to be made is what work will be done on which data. In case each process is performing a different task on different data we are talking about a MPMD (Multiple Program Multiple Data) programming model. If on the other hand the same program is running on every processor but with different data, this corresponds to a SPMD (Single Program Multiple Data) programing model. In this project we decided to use the SPMD paradigm.

Domain decomposition and load balancing techniques are a very suitable way of decomposing data and work in numerical analysis. Using load balancing tools like those of *Jostle[3], Metis [4]* or *ParMetis* the original computational domain can be divided into a number of subdomains, equal to the number of available computing units, and these subdomains can be distributed to the processors. In this way the resulting linear systems become smaller compared to the global one. Generally, the two CPU time-consuming parts of a finite element program are the calculation of the entries in the system matrices (element loop) and the solving of the resulting linear systems.

In particular, the calculation of the system matrices is done element-wise without any dependency on the computation between different elements. Therefore, the contributions of different elements can be calculated on different processors. When solving the linear system of equations additional care has to be taken for the unknowns residing on the artificial boundaries.

5 GS/RF Performance

We tested GS/RF on the SX6 with an artificial 2D test model consisting of 10620 elements.

The fastest solver used to solve the linear system of equations is the BiCGSTAB method. Applying the original non-optimized code took 3412.62 seconds and achieved 21.2 MFlops on one NEC SX6 processor. A more detailed analysis of the relevant GS/RF code showed that the most CPU time-consuming part of the BiCGSTAB algorithm is the matrix-vector multiplication. 99.7 % of the CPU time (3403.294 sec) is spent in computing the matrix-vector product.

The available NEC compilers offer directives which support code vectorization. Employing some of these directives the same test case terminates after only 74.04 secs, thus achieving a performance of 1015.1 MFlops. The overall

Table 1. Comparision of non-vectorized and vectorized code versions

	non-vectorized version		**vectorized version**	
	total	matrix-vector	total	matrix-vector
Computation time [secs]	3412.62	3403.29	74.04	63.46
Performance [MFlops]	21.2	21.2	1015.1	1182.6

time spent in the matrix-vector multiplication of the vectorized code is only 63.46 seconds at a rate of 1182.6 MFlops. Table 1 summarizes these performance measurements.

Fig. 4 depicts the resulting pressure distribution between the top right and bottom left corner of the computational domain calculated on the SX6.

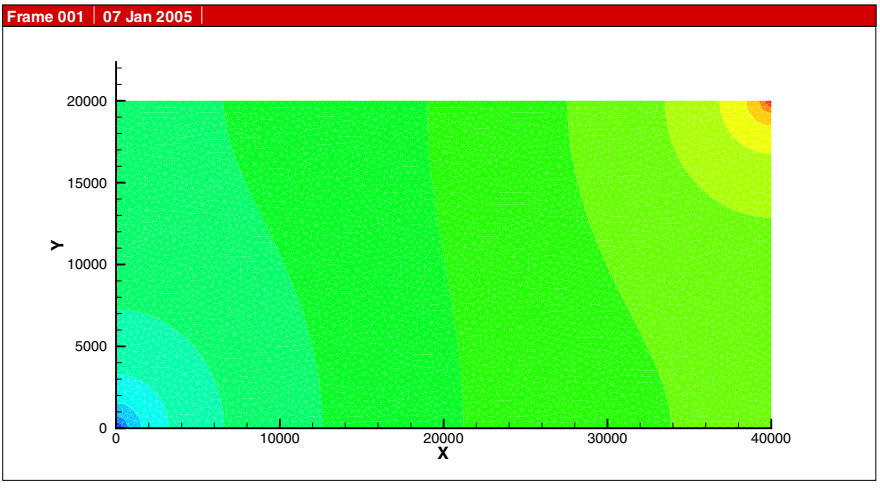


Fig. 4. Pressure distribution between top right and bottom left

Not all codes are suitable for vectorization. Our results presented above show that the GS/RF code is suitable for vectorization.

Our real-world target models consist of hundreds of millions of elements. Thus, in order to take advantage of the architecture of the NEC SX6 and the SX8 respectively and to achieve higher performance, it is necessary to parallelize the code. The next step of our project is therefore to implement a parallel version of the BiCGSTAB algorithm based on domain decomposition and load balancing techniques. In a later stage GS/RF should be imbedded either in the *Globus* [5] or *UNICORE* [6] environment. For parameter studies the usage of *Condor* [7] seems to be very interesting.

6 Conclusion

In this work we have presented a geo-process software system, modeling complex coupled processes found in geo-systems. The results of the simulation of the FEBEX problem, a full scale engineering barriers experiment in crystalline rock for high level radioactive waste (HLW) repository show that the use of conventional hardware impose limitations on reliable modeling and simulation due to computation time.

A GRID environment can provide and enable the exploitation of the necessary resources to overcome the above mentioned limitations. As shown in this paper, vectorization of the serial GS/RF code leads to performance gains which is the first step in porting the code to high performance computing architectures. Further work on the GS/RF code will be done on parallelizing the solver algorithm alongside the coupling of shallow water and groundwater flow, as well as including kinetic biogeochemical reactions as well as improving the multi-view and three-dimensional graphics for the GUI.

References

1. GeoSys/RockFlow Team: GeoSys/RockFlow Web Page (2005) http://www.uni-tuebingen.de/zag/geohydrology/index.html.
2. Wang, W., Mingliang Xie, O.K.: FEBEX Benchmark Studies. GeoSys - Preprint 2004-40, Center of Applied Geosciences, University of Tübingen (2004)
3. Walshaw, C., Cross, M.: Parallel Optimisation Algorithms for Multilevel Mesh Partitioning. Parallel Comput. **26** (2000) 1635–1660 (originally published as Univ. Greenwich Tech. Rep. 99/IM/44).
4. Karypis, G., Kumar, V.: METIS:A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Technical report, University of Minnesota,Department of Computer Science / Army HPC Research Center (1998)
5. Foster, I., Kesselmann, C.: The Globus Project: A Status Report. In: IPPS/SPDP'98 Heterogeneous Computing Workshop. (1998) 4–18
6. (Ed.), D.E.: Joint Project Report for the BMBF-Project UNICORE Plus. ISBN: 3-00-011592-7 (January 2000 - December 2002) Grant Number: 01 IR 001 A-D.
7. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Condor – a distributed job scheduler. In Sterling, T., ed.: Beowulf Cluster Computing with Linux. MIT Press (2001)

H2O Metacomputing - Jini Lookup and Discovery

Dirk Gorissen, Gunther Stuer, Kurt Vanmechelen, and Jan Broeckhove

Antwerp University, Middelheimlaan 1, 2020 Antwerp, Belgium
Dirk.Gorissen@student.ua.ac.be,
{Gunther.Stuer, Kurt.Vanmechelen, Jan.Broeckhove}@ua.ac.be

Abstract. Because of its inter-organisational, collaborative use of computational resources, grid computing presents a severe interoperability challenge to grid application developers. Different middleware technologies need to be bridged in order to fully utilise the power the grid provides. This paper describes a bridge between two such middlewares: The H2O Metacomputing Framework and Jini technology. The paper details how H2O resources may be registered, discovered and used as Jini[1] services. Both technologies are introduced, design decisions discussed and a fully functional implementation presented.

1 Introduction

The term *Grid* refers to an infrastructure that enables the integrated, collaborative use of (high-end) computers, networks, databases, and scientific instruments owned and managed by multiple organisations [1]. Consequently, by definition, grid computing implies collaboration and coordination across organisational and geographical boundaries. This increased scale in turn means that the environment we are dealing with is increasingly heterogenous and that interoperability becomes a necessity rather than a feature. This interoperability manifests itself at three levels: the underlying network architecture, the middleware, and the administrative and organisational structure.

This paper will concentrate on the middleware. It will describe a bridge between two grid technologies: Emory University's H2O Metacomputing Framework [2] and Sun Microsystems' Jini technology [3]. More concretely we will describe a way to automatically publish H2O resources as Jini services in order to extend the reach and flexibility of the resulting middleware.

We start off with an introduction to both H2O and Jini in sections (2) and (3) followed by a brief discussion of their integration in section (4). Section (5) then goes on to analyse the different design decisions that need to be made while section (6) describes the actual implementation itself. We conclude the paper in section (7) with a summary and an outline of future work.

2 H2O

H2O is developed by Emory University and is a novel component-based, service-oriented framework for distributed metacomputing. Adopting a provider-centric view

[1] Jini and Java are registered trademarks of Sun Microsystems.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1072–1079, 2005.
© Springer-Verlag Berlin Heidelberg 2005

of resource sharing it provides a lightweight, general-purpose, Java based, configurable platform. H2O adopts the microkernel design philosophy: Resource owners host a software backplane (called a *kernel*) onto which owners, clients, or third-party resellers may load components (called *pluglets*) or component-suites that deliver value added services without compromising owner security or control [2]. For easy identification every H2O resource (kernel or pluglet) is identified by a unique URL (eg: http://myKernel:7799/).

The relationship between a kernel and its pluglets is best understood in terms of an analogy with servlets. Servlets are Java entities used to perform cgi-like processing tasks and need to be deployed within a servlet-container (eg: Tomcat from the Apache Jakarta project). In the same way as Tomcat is a container for servlets, H2O is a container for services, aka pluglets.

H2O was designed to cater for lightweight, general purpose, loosely coupled networks where resources are shared on a P2P basis. Consecutively, H2O does not target large scale computer networks (like Globe [4] does) nor does it aim for adoption in high speed, performance critical networks (like TAO [5]).

While the kernel is completely under the resource owner's control, authorised clients may configure, securely use and even freely deploy new pluglets into the kernel. It is this last aspect that makes H2O unique. Namely, by separating the service provider from the resource owner it becomes possible for any authorised third party to deploy new services in a running kernel.

3 Jini

Jini was developed by Sun Microsystems in the late 1990's and many of the people who created and championed it are the same people who originally created and championed certain aspects of Java. Jini is a distributed, service-oriented system based on the idea of federating groups of users and the resources required by those users. Jini's overall goal is to turn the network into a flexible, easily administered tool on which resources can be discovered and used by human and computational clients.

Jini services are represented as Java interfaces and are registered with the Jini *Lookup Service (LUS)*. The LUS plays the role of the service broker and is the bootstrapper of a Jini federation. It is automatically discovered through the *discovery* protocol (spontaneous computing) and its services are queried through the *lookup* protocol. Jini further supplies a distributed transaction and event manager, pluggable transports, an extensive (mobile code) security framework and a self-healing mechanism based on leasing. The latter means that each service is granted a lease when it joins the Jini federation and it is up to the service to renew this lease as long as it wants to remain part of the federation. In this way zombie services are automatically removed. All this makes Jini a very robust platform for grid computing, one that safely accommodates the *Eight Fallacies of Distributed Computing* [6].

A big advantage of Jini is its flexibility. Jini is basically a set of APIs that define how services connect to one another and not what those services are, what they do or how they work. Applications of Jini technology range from autonomous robotic swarms [7], grid middleware (JGrid [8, 9], ICENI [10], JISGA [11]), HPC [12], to online air ticket reservation systems (Orbitz.com).

Jini also enjoys a large community with numerous projects that extend the reach and capabilities of the Jini framework. Examples include Judy [13] (integration of Jini and Webservices) and Rio [14] (provides dynamic service containers, monitoring, load distribution, QoS enforcement and Resource Cost analysis mechanisms).

4 Building Bridges

The features and properties of Jini such as spontaneous, ad hoc networking, failure resilience, service discovery and the service-oriented programming model make it a very suitable base for creating dynamic, reliable grid systems. Despite this fact, little work has been done in creating Jini-based grid systems. This is mainly due to a number of persistent misconceptions, ie. Jini is just for devices, it is Java only, it wont work with webservices, it is not secure, Jini is tied to RMI, etc. All of which have been proven to be wrong [15].

While both H2O and Jini provide service-oriented resource abstraction layers, the major difference between the two is the conceptual approach. H2O strives to group services into one, homogeneous container. This provides a uniform, well defined paradigm for services to operate in, making tasks such as authentication and resource access easier to manage. In contrast, Jini is much more distributed and loosely coupled in nature, placing no restrictions on how services are deployed. Another difference is that, in addition to the resource abstraction, Jini provides discovery and registration protocols to enable dynamic, spontaneous computing.

It should now become clear that both technologies perfectly complement each other. Jini excels in exactly that what H2O lacks (a registration and discovery scheme) and vica versa. We will now go on to describe how H2O can be extended with Jini registration and discovery support. In this way a prospective H2O user no longer needs a priori knowledge of the location of H2O resources (kernels and pluglets).

5 Design Decisions

Before we outline how we implemented such registration support, a discussion concerning the different design decisions is in order.

Our investigation will be subdivided into four sections: the 'What', the 'Who', the 'When' and the 'How'.

5.1 What to Register

The first question we need to ask is what do we want to register. Simply the kernel, or the individual pluglets as well? The second choice results in each pluglet receiving a separate entry in the LUS that must be kept up to date. The complications that arise from this are discussed later on. The other option would be to only register the kernel and keep the pluglets safely hidden. A discovery wrapper class could then be used to hide kernel access methods, effectively making pluglet discovery transparent for the client. However, this mechanism does introduce significant overhead and restricts the

discovery of H2O services to clients with H2O access only (the wrapper will require H2O login credentials). We therefore opt for the first method.

5.2 Who Will Do the Registering

Next we have to ask ourselves who will ultimately bear the registration burden. This is important since different entities have access to different information and thus have different registration powers. We will consider 3 candidates: The kernel itself, a pluglet, or any interested third party.

The first is not desirable since it requires altering the kernel codebase and would break the H2O microkernel philosophy. A kernel and its pluglets should be unaware of any registration scheme.

Another possibility pushes the registration responsibilities completely away from H2O, towards any interested third party with enough information to do the job. For example: any Jini knowledgeable entity who knows the kernel or pluglet URL is free to register this information in the Jini lookup service. A client with login access to H2O could do even more: it can obtain a kernel/pluglet reference and register this together with the URL. However, registering a live kernel reference would allow unrestricted access to the H2O kernel, something that should be avoided. This options raises some leasing questions as well. Namely, how will the third party guarantee that the proxy it registers actually represents a live kernel?

The third option can be seen as a mixture of the above two. The idea is to let a designated registration pluglet do the registering. This way we don't have to litter the kernel with registration code and still maintain some degree of control. The lease problem is also solved though it involves a little more work. Concentrating all registration services in one designated pluglet means that this one pluglet is then responsible for handling all Jini leases. This implies that:

1. The registration pluglet needs to keep track of pluglet state changes and all the leases it holds so far. When a pluglet is destroyed the registration pluglet needs to cancel the relevant lease.
2. If the registration pluglet dies or is destroyed for some reason all the other pluglets are automatically removed from the LUS since their leases are no longer renewed.

(1) is possible through some extra coding and we we assume (2) to be unlikely.

5.3 When to Register

A third, overlapping question is "When should H2O resources be registered?". To answer this we have to discriminate between kernels and pluglets. We commence with the kernel case, kernel registration can occur on startup or at run time.

The option that comes to mind first is to register the kernel automatically on startup. The only sensible way to achieve this is through a registration pluglet that is automatically deployed during the kernel bootstrap process. The pluglet then performs the registration as soon as it gets loaded during kernel startup.

The other option is to register at runtime. Again a registration pluglet is used but the kernel now has to be registered through a manual call to a method like `regKernel()`.

Registering pluglets is handled similarly. Pluglets can be registered at deployment time or at some later stage in their life. A registration pluglet can take care of both cases by (1) listening for kernel generated pluglet deployment events and performing the registration when they occur or (2) by providing a method
like `registerPluglet(String name)` that can be manually invoked.

5.4 How to Register

Having decided who will do the registration and at what time, we can now consider how the actual registration will take place. We list the different possibilities.

Resource URL. The easiest option is to simply use resource URLs. Each H2O kernel and pluglet is identified by a unique URL (eg: http://myhost:7799/myPluglet). Registering these means that the client no longer needs a H2O URL on startup, it can discover one itself. A wrapper class can be written that discovers a LUS, finds the appropriate H2O service (kernel or pluglet) and uses the thereby obtained URL's to perform a H2O login.

Though easy to implement, this approach does have a number of downsides:

- A bundle of URLs contain no information about the underlying kernel. Although extra meta data can be attached in the form of Jini `Entry` objects this metadata is constrained to being semi-static in nature (eg: version, name, application domain, underlying platform, etc). More interesting, faster changing attributes such as load, available pluglets, etc would require an additional login step first.
- It is impossible to perform service interface based lookups, the standard way of discovering services in a Jini system.

Raw Remote Reference. Completely orthogonal to registering a URL is registering a live, Java, remote reference which points back to a kernel/pluglet. The obvious objection here, among others, is that distributing such a reference gives anybody, free, unconditional access to the kernel. Effectively bypassing every H2O security mechanism.

Resource URL & Service Interfaces. This option intends to be a compromise by ensuring that the registered proxy provides the resource URL and 'implements' every service interface. The catch is that the proxy only provides the interface, but not its implementation (invocations would throw exceptions). This 'empty box' approach allows interface based lookup while forbidding unrestricted access.

Resource URL & Dynamic Information. A downside of the URL approach is that the resulting proxy tells us nothing about the kernel it represents. It would be nice if a client could choose a particular kernel based on dynamic attributes such as load, number of pluglets, etc. Enabling this requires registering the kernel under an interface that, besides the kernel URL, provides methods to retrieve such dynamic information. The

implementation of these methods would be through a remote stub that, in accordance with the microkernel design philosophy, calls back to a pluglet that provides dynamic kernel monitoring. In this case the proxy that is registered is called a *smart proxy* since it is more than just a simple stub. Note that this option assumes such dynamic information is allowed to be freely accessed. This option can also be combined with the previous one.

Login Enabled Smart Proxy. A final option is to extend the smart proxy above with a method that logs into the kernel the proxied resource belongs to without requiring an explicit resource URL, as is usually the case. In practice this means that, in the kernel case, the client would first lookup a proxy representing a kernel and then use it to log into the H2O system by calling `getKernelContext(Credentials)`. An explicit URL parameter is no longer needed since the proxy contains one.

In the pluglet case the proxy additionally implements all functional interfaces (thus allowing service interface based lookup) but only allows access to them if a login has been performed first. This would effectively make the whole H2O system completely transparent.

6 Proposed Implementation

Since we are working within a plug-in based paradigm the most natural approach is to encapsulate all the registration logic into a separate pluglet and deploy it within a H2O kernel. The goal is to have the pluglet, on deployment or when triggered, detect which kernel it finds itself in and register it with a Jini LUS. The same goes for registering pluglets. In this way any H2O kernel can easily be made 'Jini-aware' by simply deploying a standard registration pluglet. Regarding the proxy architecture, we opt for a configurable approach instead of forcing one particular method on the user. Depending on the method invocation handler used to generate the proxy either the "Empty Box" or "Login Enabled Smart Proxy" architecture is used. Thus it is the invocation handler object that ultimately defines the behaviour of the proxy. Besides the two implementations described above users are free to define their own invocation handler in order to customise the way method invocations are handled.

The functioning of the registration pluglet is as follows (*see* figure 1):

1. At deployment time, start the discovery process, register a proxy to the kernel in every discovered LUS and listen for pluglet deployment events.
   - To register the kernel the pluglet first looks to see if there is a monitor pluglet available. If so, it is incorporated into the resulting proxy.
2. As soon as a deployment event arrives, obtain all relevant information from the newly deployed pluglet.
3. Use this information to generate a proxy, supporting all functional interfaces, using the specified invocation handler.
4. Register the object with the relevant Jini LUSs, and add the lease to the set of managed leases.

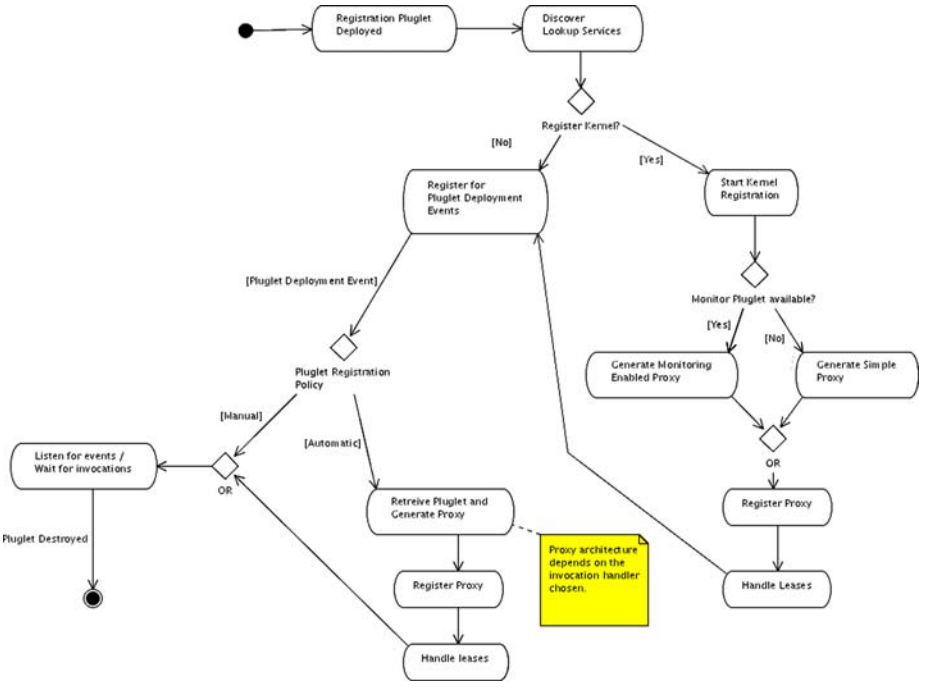


Fig. 1. Pluglet Activity Diagram

Note that if the registration pluglet receives a pluglet destroyed event, it will remove the corresponding lease form its lease map again.

In addition to the above an *AutoPopulateNewLus* switch further makes sure that newly discovered LUSs are automatically populated with H2O resources.

Source code and documentation of the pluglet is freely available at: http://h2j.jini.org.

The pluglet described here is currently in use at the UA to maintain information about available kernels and pluglets. This information is then used by grid application developers and users to find available resources. For example, the registration pluglet is currently being used within the context of an application that simulates the quantum physical three cluster problem.

7 Summary and Future Work

In this paper we have demonstrated how we can use Jini to extend the H2O Metacomputing Framework with discovery and registration support. In this way H2O becomes usable in a real world settings since clients no longer need a priori knowledge of service URLs. An implementation of a Jini registration pluglet was described together with the relevant design decisions.

Future work in this area will concentrate on adding features to the registration pluglet, developing a pluglet that encapsulates core Jini services (LUS, activation daemon, event mailbox, ...) and on Jini-based H2O resource management in general. An interesting avenue that may be pursued is the attachment of GUIs to each registered proxy to enable remote administration and enhance the client-side experience. This would be possible through the Jini ServiceUI infrastructure which allows one to add complete user interfaces to Jini services without any prior client-side installation.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Lecture Notes in Computer Science **2150** (2001)
2. Sunderam, V., Kurzyniec, D.: Lightweight self-organizing frameworks for metacomputing. In: Proc. of 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02), Edinburgh, Scotland. (2002) 119–122
3. Edwards, K.W.: Core Jini. 2 edn. Prentice Hall (2001)
4. Homburg, P., van Steen, M., Tanenbaum, A.: An architecture for a wide area distributed system. In: Proc. of 7th ACM SIGOPS European Workshop, Connemara, Ireland. (1996) 75–82
5. Schmidt, D.C., Levine, D.L., Mungee, S.: The design of the TAO real-time object request broker. Computer Communications **21** (1998) 294–324
6. Deutsch, P.: The Eight Fallacies of Distributed Computing (2004) http://java.sun.com/people/jag/Fallacies.html.
7. Hawthorne, C.: Application of Jini technology to a Swarm of Autonomous Vehicles (2003) Presentation at 7th Jini Community Meeting http://www.jini.org/meetings/seventh/J7abstracts.html#swarm.
8. Juhasz, Z.: JGrid: Jini as a Grid technology. IEEE TFCC Newsletter (2003)
9. Juhasz, Z., Andics, A., Pota, S.: JM: A Jini Framework for Global Computing. In: Proc. of 2nd International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems at IEEE International Symposium on Cluster Computing and the Grid (CCGrid'2002), Berlin, Germany. (2002) 395–400
10. Furmento, N., Lee, W., Mayer, A., Newhouse, S., Darlington, J.: ICENI: An Open Grid Service Architecture Implemented with Jini. In: Proc. of SuperComputing 2002, Baltimore, USA. (2002)
11. Y.Huang: JISGA: A Jini-based Service-oriented Grid Architecture. The International Journal of High Performance Computing Applications **17** (2003) 317–327 ISSN 1094-3420.
12. Ye, R., Hsu, W., Liu, Z., Vee, V.: Jini-enabled high performance computing. In: Proc. of 15th International Conference on Information Networking (ICOIN'01), Beppu City, Japan. (2001) 145–152
13. Asberry, D.: The judy project, seamless integration of Jini with Webservices (2004) http://judy.jini.org.
14. Reedy, D.: Project rio: A dynamic adaptive network architecture. Technical report, Sun Microsystems (2004)
15. Creswell, D.: Jini Technology, Setting the Record Straight (2004) http://www.dancres.org/jini_setting_straight.html.

User Experiences with Nuclear Physics Calculations on a H2O Metacomputing System and on the BEgrid*

P. Hellinckx, K. Vanmechelen, G. Stuer, F. Arickx, and J. Broeckhove

Department of Mathematics and Computer Science,
University of Antwerp, BE-2020 Antwerp, Belgium
peter.hellinckx@ua.ac.be

Abstract. We report on user experiences gathered with quantum nuclear scattering calculations on a H2O based metacomputing system and on the BEgrid, the Belgian EGEE Grid infrastructure. We compare quantifiable aspects such as Grid utilization but also ease-of-use and the experience encountered in porting the application to each of the Grid environments.

1 Introduction

While Grids are becoming a fact of life in High Performance Computing (HPC) it is clear that the term stands for a whole range of middleware systems with strongly varying characteristics. Although traditional Grid systems often host an extensive stack of Grid services, many applications do not necessarily require explicit coordination and centralized services for authentication, registration, and resource brokering. For these applications, a lightweight model in which individuals and organizations share their superfluous resources on a peer-to-peer basis, seems to be more suitable [1].

In order to test this claim, we have implemented a quantum physics, nuclear 3-cluster scattering problem using both a lightweight and a heavyweight Grid. We chose this test case, of which the details will be described later, for several reasons. First, the computation has both a CPU-intensive and a data-intensive part. Secondly, the computation is too extensive to solve on a single machine, yet it does not require the full blown Grid infrastructure. Thirdly, this application fits an algorithmic approach that is rather common in many scientific computations.

This contribution presents the results of this comparative study. Our choice for a lightweight Grid system is H2O [2], as we have considerable experience using this middleware. Our choice for a heavyweight Grid is the BEgrid which hosts the Belgian part of the EGEE project. As a member of the BEgrid consortium, our research group has experience in its procedures and maintenance,

* Research supported in part by the *Ministerie van de Vlaamse Gemeenschap, Afdeling Technologie en Innovatie* as part of the *Vlaamse pilot-gridinfrastructuur* project.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 1080–1088, 2005.
© Springer-Verlag Berlin Heidelberg 2005

as well as in gridifying applications. In what follows, the design, experimental results and a performance comparison between both platforms will be discussed.

2 Problem Statement

To obtain physical properties of quantum systems, such as atoms, nuclei or molecules, one needs to solve the so-called Schrödinger equation. In order to solve it, proper boundary conditions must be chosen. The solutions, the so-called "wave-functions" then allow for the calculation of physical quantities. The equation and its boundary conditions are usually too complex to be solved for many-body systems (e.g. a nucleus), and approximations have to be introduced. One approach is to expand the wave-function on a discrete, infinite-dimensional, set of basis states. Substitution of this approximation in the equation and boundary conditions, leads to a much simpler matrix equation in the expansion coefficients to be solved.

The matrix formulation can be further simplified by choosing expansion bases with specific properties. The Modified J-Matrix (MJM) model [3] is such an approach, and has been applied to 3-cluster nuclear systems [4]. We consider it here to obtain scattering results for a 3-particle configuration of a triton and 2 neutrons of ^5H. The calculations essentially consist of two steps: (1) a CPU-intensive calculation of the matrix elements in the matrix equation, and (2) the solution of the matrix equation. Step 2 can be obtained in reasonable time on a single node, and we will therefore only consider the gridification of step 1.

The oscillator expansion basis for the solution of a 3-cluster problem, is enumerated by a set of indices. These are the hypermoment K, describing the three cluster geometry; relative angular momenta l_1 and l_2 between the three clusters coupled to the total angular momentum L, a constant of motion; the oscillator index n. The number of $(l_1 l_2)$ combinations depends on both K and L. The essential matrix to be determined is the energy matrix, denoted by

$$\left\langle K_i, (l_1 l_2)_i L, n_i \left| \hat{H} \right| K_j, (l_1 l_2)_j L, n_j \right\rangle = \left\langle K_i, l_i, n_i \left| \hat{H} \right| K_j, l_j, n_j \right\rangle \qquad (1)$$

where $\hat{H}$ is the Hamiltonian, or energy, operator, and i and j distinguish basis states; the right hand side in (1) simplifies the notation, omitting L as an overall constant, and replacing the combination $(l_1 l_2)_i$ by l_i.

The theory to obtain (1) [4] is well beyond the scope of this paper, but it can be broken down to

$$\left\langle \left\langle K_i, l_i \left| \hat{H} \right| K_j, l_j \right\rangle \right\rangle = \sum_t \sum_{l_r} \sum_{l_s} R(K_i, l_i, l_r, t) \left\langle \left\langle K_i, l_r \left| \hat{H} \right| K_j, l_s \right\rangle \right\rangle_t R(K_j, l_s, l_j, t) \qquad (2)$$

where $\langle\langle\rangle\rangle$ stands for a matrix over all n_i, n_j indices. The R factors are so-called Raynal-Revai coefficients as discussed in [4], and the nature and range of index t depends on the nucleus (^5H) and its cluster decomposition ($t+n+n$).

The granularity of the problem is clear from (2), and reduces the problem to a fork and join algorithm by calculating all independent $\left\langle\left\langle K_i, l_r \left| \hat{H} \right| K_j, l_s \right\rangle\right\rangle_t$ matrices for fixed K_i, K_j and all allowed combinations l_r, l_s, t (the fork), followed by a summation to obtain (2) (the join). In this paper we discuss a calculation for $L = 0$ and $K = 0, 2, 4, .., 16$, a range of $l_1 = l_2 = 0, 1, .., K/2$ values, and 45 t values. All of the computational code components are implemented in Fortran90 using the Intel v7 compiler.

A farmer-worker distribution algorithmic model seems an evident choice for this problem, because all of the fork tasks are independent of one another. The tasks are file based, meaning that they get their input from a series of files and write their results into one. All input files except one, a configuration file which contains the particular indices for the current computation, have a constant content for all tasks.

3 H2O

H2O [2] is developed by Emory University and is a component-based, service-oriented framework for distributed metacomputing. H2O is designed to cater for lightweight, general purpose, loosely coupled networks where resources are shared on a P2P basis. Adopting a provider-centric view of resource sharing, it provides a lightweight, general-purpose, Java based, configurable platform. H2O adopts the microkernel design philosophy: resource owners host a software backplane (called a kernel) onto which owners, clients, or third-party resellers may load components (called pluglets) that deliver value added services. It is this last aspect that makes H2O unique: by separating the service provider from the resource owner, it becomes possible for any authorized third party to deploy new services in a running kernel.

3.1 Software Architecture

Since H2O basically stands for an empty service container, no infrastructural Grid components exist. In order to keep the H2O approach as lightweight as possible, we implemented the infrastructural components which are absolutely necessary: a lookup service (LUS), a file transfer pluglet (FTPl), a broker, and executors. Figure 1 shows the basic architecture of our framework.

Before the calculation can start, the infrastructure has to be set up. This comes down to starting the LUS, and initiating the H2O kernel on all participating nodes. These kernels will register themselves with the LUS and deploy the FTPl. From this point on, the system is ready to start the calculation.

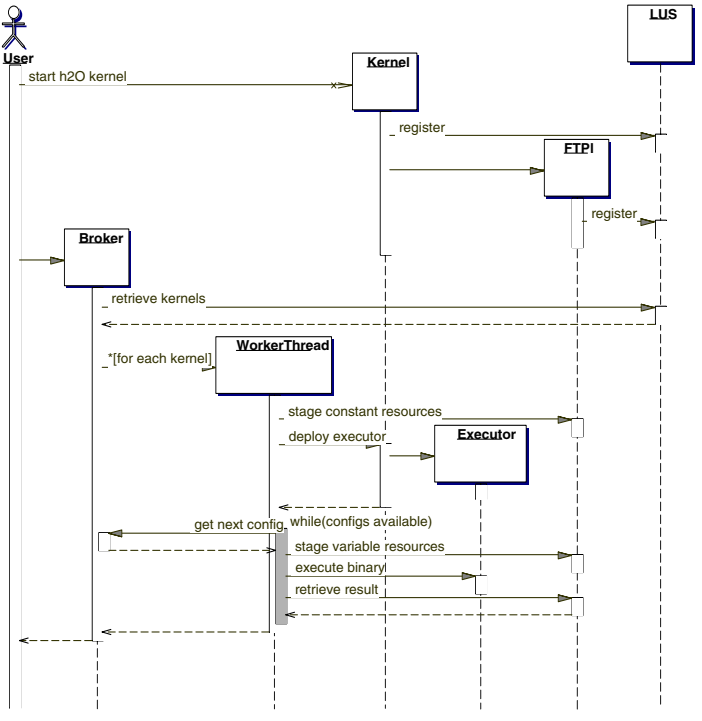


Fig. 1. Architecture of the framework

The first step in distributing the problem involves initializing the broker component which is responsible for the overall progress of the calculation. It will locate all available kernels using the LUS and instantiate *WorkerThread*s for all of them. Each one is responsible for the communication with one particular kernel. They start by staging the constant-content input files and the Fortran binary using the FTPl, followed by deploying the executor pluglet. Next, as long as there are sets of indices left to calculate, they will generate configuration files, transfer them to the kernel, start the executor, and retrieve the result. Finally, when all required combinations have been calculated, this process stops and the results can be recombined.

4 BEgrid

The BEgrid project involves an infrastructural collaboration between several universities and research institutions in Belgium. It is currently running LCG 2.2 middleware [5] and includes sites from the universities of Antwerp, Ghent, Leuven and Brussels as well as other research oriented institutions such as the Flanders Marine Institute. More institutions are expected to join on a short term

basis. BEgrid is part of the North European ROC region of the EGEE project. Many sites are currently in deployment phase, 205 CPUs will be operational in the beginning of 2005.

4.1 Software Architecture

The LCG middleware comprises of a number of functional modules that cooperate to provide workload, data and information management services. A *resource broker* matches job requirements, which are expressed in JDL (Job Description Language), to the available resources. The user has the ability to state complex infrastructural and operational job requirements using a standardized information schema called GLUE. The broker also supports user-defined ranking of resources that fulfill the hard job requirements, based on soft QoS requirements such as the average queued time for a job at a compute resource. After a computing resource is selected by the broker, the job is forwarded to the *computing element* (CE) associated with that resource. The computing element virtualizes compute resources such as PBS, LSF or Condor farms, providing uniform access to heterogeneous compute back-ends.

In the same way as the CE provides a virtualization layer for compute resources, a *storage element* (SE) delivers such a layer for heterogeneous storage resources. A *replica management service* (RMS) supports file based data replication and orchestrated access to files stored across SEs. Although jobs may interact with the RMS to access large data files, small I/O requirements can be fulfilled by specifying input and output *sandboxes*. These allow the user to indicate which files should be staged in and out before and after the computation. All interactions with the Grid infrastructure take place from a *user interface* node which supports APIs, command line, and graphical interfaces to BEgrid.

The LCG middleware supports a batch oriented computing model, and is currently in process of providing better support for data dependencies between jobs and more tightly coupled computations. This does not affect the mapping of our problem to the LCG based computing infrastructure as jobs are trivially parallel. A characteristic that does affect our job distribution scheme is the fact that long delays have been observed when submitting jobs and receiving their results on a zero loaded computing resource. The computation time for individual indices also varies widely from a few seconds to a few hours. Therefore it was deemed necessary to group multiple calculations in a single job, in order to mitigate the performance penalty induced by these delays. To determine the size of these groupings we monitored the number of queued jobs over all CE queues during the computation and adjusted the group size accordingly. The idea is that as long as all CE queues are sufficiently loaded to keep the compute resources busy, the submission delay will not affect the total wall-clock time of the computation.

All file staging was performed using the input and output sandboxes since the calculations are compute intensive. After jobs finish, a polling component on the user interface fetches the job output.

5 Experimental Results

5.1 Speedup

The distributed computations on the H2O framework were performed on a local cluster containing a heterogeneous set of 20 PC-nodes with an Intel processor ranging from 1.7 to 2.66 GHz, part of a 100Mb LAN. For the BEgrid tests, we used 24 3.2 GHz Pentium 4 nodes at Antwerp and 8 dual-Opteron/244 nodes (1.8 GHz) at Leuven. This amounts to 33 normalized 3.2 CPUs which is the maximum reachable speedup factor on this setup. Both sites use 1 GigE interconnect. The user interface and resource broker were both located at Antwerp.

Table 1 shows the turn-around time of the distributed computations for different values of K, and compares these to the pure sequential execution timing. The speedup factor which denotes the ratio between sequential and distributed execution time is also shown for each K. The table depicts low speedup figures for computations with $K < 10$ both for H2O and BEgrid. More time is spent deploying the jobs than processing them. The reason is that most of the tasks with low K have a very small execution time. From $K = 10$ onwards, the picture changes, as the mean execution time for these jobs rises significantly. The results are almost optimal for $K > 12$ on H2O. For H2O, the low speedup for small K is a direct consequence of our naive implementation of the Grid infrastructure components. Prefetching the data files will probably boost speedup for small K.

For BEgrid, low speedup values can be attributed to job submission delays which are in the order of seconds, and result fetching delays which are in the order of minutes. These make it difficult to achieve optimal resource utilization in the beginning and at the end of the computation. The delays were not found to be caused by data transmission overhead but rather by the workload management system itself.

Table 1. Speedup achieved on the available H2O and LCG resources

	H2O			LCG		
K	sequential	distributed	speedup	sequential	distributed	speedup
0	2s	55s	0.04	1s	7m57s	0.00
2	9s	1m26s	0.11	3s	9m00s	0.01
4	1m11s	2m24s	0.49	35s	16m27s	0.04
6	9m01s	4m00s	2.25	5m19s	16m59s	0.32
8	56m31s	8m25s	6.72	30m44s	18m54s	1.63
10	4h52m56s	20m22s	14.38	2h55m43s	18m42s	9.40
12	21h33m12s	1h12m52s	17.75	12h37m12s	49m36s	15.27
14	82h38m21s	4h11m24s	19.72	49h39m20s	2h01m05s	24.61
16	285h04m20s	14h23m59s	19.80	173h37m26s	6h07m31s	28.30

5.2 Utilization

Turn-around timing is not an acceptable measure if we wish to compare distributive platforms, such as the H2O and LCG middlewares, as the amount of

available resources varies significantly from one run to another. Therefore, we consider the utilization rate of the participating nodes, which is defined as the percentage of time the processor is performing the actual computation. This measure provides an indication of the overhead introduced by both the middleware and data transfers, and is largely independent of the hardware used. Figure 2 shows the utilization as a function of K, which indicates strongly decreasing overhead for increasing K-values. For H2O, the main part of the overhead can be attributed to data transfers. The remaining part, 0.03% to 0.07% of the total turn-around time, is middleware overhead. For BEgrid, job submission and result fetching delays were the main causes for overhead.

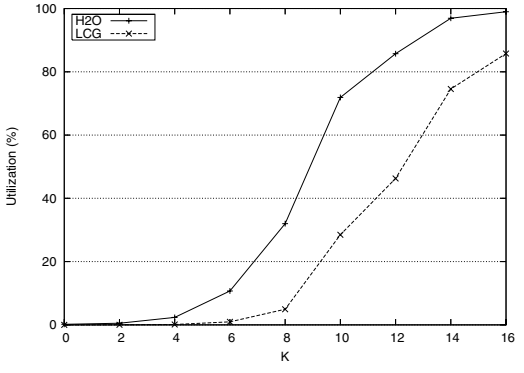


Fig. 2. Processor Utilization Comparison between H2O and BEgrid

6 H2O vs. BEgrid

The main merits of H2O are that it is lightweight and easy to deploy. As it is Java based, it functions on any desktop commodity PC available, limiting hardware investments. Furthermore, job submission is very responsive, which allows for the submission of small jobs, without creating too much overhead. Finally, its component based architecture allows for on-demand deployment of custom made, and highly optimized Grid components such as specialized schedulers, LUS, etc.

The latter advantage also turns out to be a shortcoming since every component has to be constructed from scratch. As the available documentation is rather scarce, this results in an initially steep learning curve. In the current implementation, we opted for a minimalistic approach which kept the infrastructural components simple, but unfortunately also unscalable and, in general, not reusable. High quality, scalable, robust, and general purpose components would require an important implementation investment. Finally, though H2O may be written in Java, typical computational components are not. Thus special provisions have to be made to ensure that kernels hosted on Linux, Solaris, Mac and Windows platforms can participate in the computation.

One of the main merits of distributing the computation on BEgrid is the efficiency of the implementation process itself: only 230 lines of bash scripting code were needed to capture all job distribution and execution logic. Though the highly parallel and independent nature of the distributed subtasks is certainly of influence, this is also directly related to the fact that core Grid services for workload, data and information management as well as core security provisions, are delivered 'out-of-the-box'. Another important advantage is the fact that coding an application to a model and infrastructure accepted on a European scale, opens the possibility for accessing a large scale infrastructure. On the other hand, one should be aware that gaining access to a general purpose (inter)national Grid resources still requires "social" interactions for the application to be accepted at remote sites.

A potential shortcoming of BEgrid is the high job submission and result fetching delay which may affect applications that have to face a fine grained level of job distribution. Although our approach of wrapping multiple application level jobs into a single Grid job mitigated these effects, it was still difficult to achieve a good load balance at the beginning and near the end of the total computation.

7 Conclusions

The main conclusion of this work is that both Grid platforms are well suited for gridifying a highly parallelizable CPU-intensive application.

In its default configuration, H2O is best suited for distributive computations on a limited scale. It is highly responsive, but only provides the base Java-oriented infrastructure to develop Grid applications. It does allow for a more scalable approach, but requires all necessary Grid components for this to be custom made. In order to overcome this drawback, a pluggable H2O based Grid infrastructure is currently under development.

BEgrid provides a platform with standardized well-defined services for distributive processing. This delimited set of services to approach the Grid is a merit of the system, because of its straightforward usage, as well as a shortcoming because of its inherent limitations. With respect to computing resources, it features international scalability, which is an important asset for large scale distributed applications.

References

1. Jonathan Chin, Peter V. Coveney: *Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware.* `http://www.realitygrid.org/lgpaper.html`.
2. V. Sunderam, D. Kurzyniec: *Lightweight Self-Organizing Frameworks for Metacomputing.* In 11th Int. Symp. on High Performance Distributed Computing, 2002.

3. J. Broeckhove, F. Arickx, W. Vanroose and V. Vasilevsky: *The Modified J-Matrix method for Short-Range Potentials.* J. Phys. A: Math. Gen 37 (2004) 1-13
4. V. S. Vasilevsky, A. V. Nesterov, F. Arickx and J. Broeckhove: *The Algebraic Model for Scattering in Three–s–cluster Systems: Theoretical Background.* Phys. Rev. C63 (2001) 034606:1–16
5. A. D. Delgado et al., *LCG-2 User Guide*. https://edms.cern.ch/file/454439/LCG-2-Userguide.pdf.

Author Index

Aloisio, Giovanni, 10
Asirvatham, Arul, 265

Bagheri, Babak, 44
Bai, Li, 273
Bao, Lichun, 485
Barsky, Brian A., 224
Basak, Tanmay, 806
Benassarou, A., 314
Bernreuther, Martin, 1
Berzins, M., 28
Beyls, Kristof, 166
Billiet, David, 1038
Bittar, E., 314
Brenk, Markus, 1
Bungartz, Hans-Joachim, 1
Byun, Daewon, 806

Cafaro, Massimo, 10
Cai, Hongming, 335
Caymes, Paola, 132
Chinchalkar, Shirish, 76
Chinnusamy, Malar, 60
Clark, Terry, 44
Coleman, Thomas F., 76
Cui, Yong, 551

D'Hollander, Erik H., 166
Dauger, Dean E., 84
Day, Mitch D., 68
Decyk, Viktor K., 84
Dobson, James E., 99

Effinger-Dean, Laura, 107
Epicoco, Italo, 10

Fabricius, Uwe, 27
Farid, Hany, 99
Freundl, Christoph, 27
Funika, Włodzimierz, 158

Gava, FrTdTric, 1038
Glasner, Christian, 124
Grama, Ananth, 664
Gyllenhaal, John, 140

Han, Kijun, 585
He, Yuanjun, 335
Heath, Michael T., 52
Hoffmann, Christoph, 664
Hoppe, Hugues, 265
Hou, Qibin, 273
Hua, Wei, 248
Hua0, Hujun, 248

Janik, Arkadiusz, 158
Jessup, E.R., 91
Jia, Jinyuan, 298
Jiang, Hong, 519
Jiao, Xiangmin, 52
Johnson, C.R., 28
Joneja, Ajay, 298
Jung, Moon-Ryul, 216

Karl, Wolfgang, 174, 182
Kim, Hyunsook, 585
Kirby, R.M., 28
Kosloff, Todd J., 224
Kozlowski, Alex, 224
Köstler, Harald, 27

Liao, Wenyuan, 798
Liu, Hua, 248
Liu, Mingzhe, 420
Liu, Qi, 368
Liu, Xinchun, 860
Liu, Zheng, 821
Loulergue, FrTdTric, 1038
Lucas, L., 314
Luque, Emilio, 132

Mansfield, Peter, 76
Marchesini, John C., 99
Margalef, Tomàs, 132
May, John, 140
Merkle, Daniel, 412
Metaxas, Dimitris, 703
Middendorf, Martin, 412
Min, Kyungha, 216
Morajko, Anna, 132
Mundani, Ralf-Peter, 1

Muntean, Ioan Lucian, 1

N. W. John, 314
Nikishkov, G.P., 232
Nishidate, Y., 232

Pakin, Scott, 149
Pan, Zhijian, 404
Pasztor, Egon, 224
Percell, Peter, 806
Perelman, Alex, 224
Praun, Emil, 265

Quarta, Gianvito, 10

Reggia, James, 404
Rendell, Alistair P., 18
Ribbens, Calvin J., 60
Ridgway Scott, L., 44
Roman, Eric, 224
Roper, James, 18
Ryu, Jungpil, 585
Rüde, Ulrich, 27

Sameh, Ahmed, 664
Sandu, Adrian, 798, 821
Scheidler, Alexander, 412
Schulz, Martin, 140
Schwarz, Susan A., 99
Sfarti, Adrian, 224
Shang, Hui, 373
Sharma, Arjun, 107
Sim, Terence, 207
Smith, Sean W., 99
Spiegl, Edith, 124
Sun, Ninghui, 860
Swaminathan, Gautam, 60

Tan, Chew Lim, 207
Tan, Guangming, 860
Tan, Haixia, 485
Tang, Kai, 298
Tao, Jie, 174, 182
Teresco, James D., 107
Thomas, Michael A., 68
Trinitis, Carsten, 191
Tsechpenakis, Gabriel, 703
Tufo, H.M., 91

Vandeputte, Frederik, 166
Varadarjan, Srinidhi, 60
Volkert, Jens, 124

Wainer, Gabriel, 368, 373
Wang, Qing, 248
Wang, Ruili, 420
Wang, Yangsheng, 273
Weidendorfer, Josef, 191
Woitaszek, M.S., 91
Woodward, Jeffrey B., 99
Wu, Jianping, 551
Wu, Yong, 335

Xu, Xian, 1029

Yang, Luobin, 68
Yang, X.S., 199
Yang, Xiaolong, 519
Yi, Ping, 593

Zeng, Weilin, 485
Zhang, Aidong, 1029
Zhang, Jian J., 199
Zhang, Min, 519
Zhang, Shiyong, 593
Zhang, Yu, 207
Zhong, Yiping, 593
Zhou, Dong, 248