

Lecture Notes in Computer Science

1610

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

G rard Cornu jols Rainer E. Burkard
Gerhard J. Woeginger (Eds.)

Integer Programming and Combinatorial Optimization

7th International IPCO Conference
Graz, Austria, June 9-11, 1999
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

G  rard Cornu  jols
GSIA, Carnegie Mellon University
Schenley Park, Pittsburgh, PA 15213, USA
E-mail: gc0v@andrew.cmu.edu

Rainer E. Burkard
Gerhard J. Woeginger
Institut f  r Mathematik, Technische Universit  t Graz
Steyrergasse 30, A-8010 Graz, Austria
E-mail: {burkard,gwoegi}@opt.math.tu-graz.ac.at

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Integer programming and combinatorial optimization :
proceedings / 7th International IPCO Conference, Graz, Austria, June
9 - 11, 1999. G  rard Cornu  jols ... (ed.). - Berlin ; Heidelberg ; New
York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ;
Tokyo : Springer, 1999
(Lecture notes in computer science ; Vol. 1610)
ISBN 3-540-66019-4

CR Subject Classification (1998): G.1.6, G.2.1, F.2.2

ISSN 0302-9743
ISBN 3-540-66019-4 Springer-Verlag 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-Verlag. Violations are liable for prosecution under the German Copyright Law.

   Springer-Verlag Berlin Heidelberg 1999
Printed in Germany

Typesetting: Camera-ready by author
SPIN: 10705123 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

Preface

This volume contains the papers selected for presentation at IPCO VII, the Seventh Conference on Integer Programming and Combinatorial Optimization, Graz, Austria, June 9–11, 1999. This meeting is a forum for researchers and practitioners working on various aspects of integer programming and combinatorial optimization. The aim is to present recent developments in theory, computation, and applications of integer programming and combinatorial optimization. Topics include, but are not limited to: approximation algorithms, branch and bound algorithms, computational biology, computational complexity, computational geometry, cutting plane algorithms, diophantine equations, geometry of numbers, graph and network algorithms, integer programming, matroids and submodular functions, on-line algorithms, polyhedral combinatorics, scheduling theory and algorithms, and semidefinite programs.

IPCO was established in 1988 when the first IPCO program committee was formed. IPCO I took place in Waterloo (Canada) in 1990, IPCO II was held in Pittsburgh (USA) in 1992, IPCO III in Erice (Italy) 1993, IPCO IV in Copenhagen (Denmark) 1995, IPCO V in Vancouver (Canada) 1996, and IPCO VI in Houston (USA) 1998. IPCO is held every year in which no MPS (Mathematical Programming Society) International Symposium takes place: 1990, 1992, 1993, 1995, 1996, 1998, 1999, 2001, 2002, 2004, 2005, 2007, 2008, ... Since the MPS meeting is triennial, IPCO conferences are held twice in every three-year period. As a rule, in even years IPCO is held somewhere in Northern America, and in odd years it is held somewhere in Europe.

In response to the call for papers for IPCO'99, the program committee received 99 submissions, indicating a strong and growing interest in the conference. The program committee met on January 10 and January 11, 1999, in Oberwolfach (Germany) and selected 33 contributed papers for inclusion in the scientific program of IPCO'99. The selection was based on originality and quality, and reflects many of the current directions in integer programming and optimization research. The overall quality of the submissions was extremely high. As a result, many excellent papers could not be chosen.

We thank all the referees who helped us in evaluating the submitted papers: Karen Aardal, Norbert Ascheuer, Peter Auer, Imre Bárány, Therese Biedl, Hans Bodlaender, Andreas Brandstädt, dan brown, Peter Brucker, Alberto Caprara, Eranda Çela, Sebastian Ceria, Chandra Chekuri, Joseph Cheriyan, Fabian Chudak, William H. Cunningham, Jesus De Loura, Friedrich Eisenbrand, Matteo Fischetti, Michel Goemans, Albert Gräf, Jens Gustedt, Leslie Hall, Christoph Helmberg, Winfried Hochstättler, Stan van Hoesel, Han Hoogeveen, Mark Jerum, Olaf Jahn, Michael Jünger, Howard Karloff, Samir Khuller, Bettina Klinz, Dieter Kratsch, Monique Laurent, Jan Karel Lenstra, Martin Loeb, Alexander Martin, Ross McConnell, S. Tom McCormick, Petra Mutzel, Michael Naatz, Karl Nachtigall, John Noga, Andreas Nolte, Alessandro Panconesi, Chris Potts, Mau-

rice Queyranne, Jörg Rambau, R. Ravi, Gerhard Reinelt, Franz Rendl, Günter Rote, Juan José Salazar, Rüdiger Schultz, Andreas S. Schulz, Petra Schuurman, András Sebő, Jay Sethuraman, Martin Skutella, Frits Spijksma, Angelika Steger, Cliff Stein, Mechthild Stoer, Frederik Stork, Leen Stougie, Éva Tardos, Gottfried Tinhofer, Zsolt Tuza, Marc Uetz, Vijay Vazirani, Albert Wagelmans, Dorothea Wagner, Robert Weismantel, David Williamson, Laurence Wolsey, Günter M. Ziegler, and Uwe Zimmermann. This list of referees is as complete as we could make it, and we apologize for any omissions or errors.

The organizing committee for IPCO'99 essentially consisted of Eranda Çela, Bettina Klinz, and Gerhard Woeginger. IPCO'99 was conducted in cooperation with the Mathematical Programming Society (MPS), and it was sponsored by the Austrian Ministry of Science, by Graz University of Technology, by the Province of Styria, and by the City of Graz.

March 1999

Gérard Cornuéjols
Rainer E. Burkard
Gerhard J. Woeginger

IPCO VII Program Committee

Gérard Cornuéjols (Chair), Carnegie Mellon University

Rainer E. Burkard, TU Graz

Ravi Kannan, Yale University

Rolf H. Möhring, TU Berlin

Manfred Padberg, New York University

David B. Shmoys, Cornell University

Paolo Toth, University of Bologna

Gerhard J. Woeginger, TU Graz

Table of Contents

<i>Market Split and Basis Reduction: Towards a Solution of the Cornuéjols-Dawande Instances</i>	1
K. Aardal, R.E. Bixby, C.A.J. Hurkens, A.K. Lenstra, and J.W. Smeltink	
<i>Approximation Algorithms for Maximum Coverage and Max Cut with Given Sizes of Parts</i>	17
A.A. Ageev and M.I. Sviridenko	
<i>Solving the Convex Cost Integer Dual Network Flow Problem</i>	31
R.K. Ahuja, D.S. Hochbaum, and J.B. Orlin	
<i>Some Structural and Algorithmic Properties of the Maximum Feasible Subsystem Problem</i>	45
E. Amaldi, M.E. Pfetsch, and L.E. Trotter, Jr.	
<i>Valid Inequalities for Problems with Additive Variable Upper Bounds</i>	60
A. Atamtürk, G.L. Nemhauser, and M.W.P. Savelsbergh	
<i>A Min-Max Theorem on Feedback Vertex Sets</i>	73
M. Cai, X. Deng, and W. Zang	
<i>On the Separation of Maximally Violated mod-k Cuts</i>	87
A. Caprara, M. Fischetti, and A.N. Letchford	
<i>Improved Approximation Algorithms for Capacitated Facility Location Problems</i>	99
F.A. Chudak and D.P. Williamson	
<i>Optimal 3-Terminal Cuts and Linear Programming</i>	114
W.H. Cunningham and L. Tang	
<i>Semidefnite Programming Methods for the Symmetric Traveling Salesman Problem</i>	126
D. Cvetković, M. Čangalović, and V. Kovačević-Vujčić	
<i>Bounds on the Chvátal Rank of Polytopes in the 0/1-Cube</i>	137
F. Eisenbrand and A.S. Schulz	
<i>Universally Maximum Flow with Piecewise-Constant Capacities</i>	151
L. Fleischer	

<i>Critical Extreme Points of the 2-Edge Connected Spanning Subgraph Polytope</i>	166
J. Fonlupt and A.R. Mahjoub	
<i>An Orientation Theorem with Parity Conditions</i>	183
A. Frank, T. Jordán, and Z. Szigeti	
<i>Parity Constrained k-Edge-Connected Orientations</i>	191
A. Frank and Z. Király	
<i>Approximation Algorithms for MAX 4-SAT and Rounding Procedures for Semidefinite Programs</i>	202
E. Halperin and U. Zwick	
<i>On the Chvátal Rank of Certain Inequalities</i>	218
M. Hartmann, M. Queyranne, and Y. Wang	
<i>The Square-Free 2-Factor Problem in Bipartite Graphs</i>	234
D. Hartvigsen	
<i>The m-Cost ATSP</i>	242
C. Helmberg	
<i>A Strongly Polynomial Cut Canceling Algorithm for the Submodular Flow Problem</i>	259
S. Iwata, S.T. McCormick, and M. Shigeno	
<i>Edge-Splitting Problems with Demands</i>	273
T. Jordán	
<i>Integral Polyhedra Associated with Certain Submodular Functions Defined on 012-Vectors</i>	289
K. Kashiwabara, M. Nakamura, and T. Takabatake	
<i>Optimal Compaction of Orthogonal Grid Drawings</i>	304
G.W. Klau and P. Mutzel	
<i>On the Number of Iterations for Dantzig-Wolfe Optimization and Packing-Covering Approximation Algorithms</i>	320
P. Klein and N. Young	
<i>Experimental Evaluation of Approximation Algorithms for Single-Source Unsplittable Flow</i>	328
S.G. Kolliopoulos and C. Stein	

<i>Approximation Algorithms for a Directed Network Design Problem</i>	345
V. Melkonian and É. Tardos	
<i>Optimizing over All Combinatorial Embeddings of a Planar Graph</i>	361
P. Mutzel and R. Weiskircher	
<i>A Fast Algorithm for Computing Minimum 3-Way and 4-Way Cuts</i>	377
H. Nagamochi and T. Ibaraki	
<i>Scheduling Two Machines with Release Times</i>	391
J. Noga and S. Seiden	
<i>An Introduction to Empty Lattice Simplices</i>	400
A. Sebő	
<i>On Optimal Ear-Decompositions of Graphs</i>	415
Z. Szigeti	
<i>Gale-Shapley Stable Marriage Problem Revisited: Strategic Issues and Applications</i>	429
C.-P. Teo, J. Sethuraman, and W.-P. Tan	
<i>Vertex-Disjoint Packing of Two Steiner Trees: Polyhedra and Branch-and-Cut</i>	439
E. Uchoa and M. Poggi de Aragão	
<i>Author Index</i>	453

Market Split and Basis Reduction: Towards a Solution of the Cornuéjols-Dawande Instances

Karen Aardal^{*1}, Robert E. Bixby², Cor A.J. Hurkens^{**3}, Arjen K. Lenstra⁴,
and Job W. Smeltink^{***1}

¹ Department of Computer Science, Utrecht University
`{aardal,job}@cs.uu.nl`

² Department of Computational and Applied Mathematics, Rice University
`bixby@caam.rice.edu`

³ Department of Mathematics and Computing Science,
Eindhoven University of Technology
`wscor@win.tue.nl`

⁴ Emerging Technology, Citibank N.A.
`arjen.lenstra@citicorp.com`

Abstract. At the IPCO VI conference Cornuéjols and Dawande proposed a set of 0-1 linear programming instances that proved to be very hard to solve by traditional methods, and in particular by linear programming based branch-and-bound. They offered these *market split* instances as a challenge to the integer programming community. The market split problem can be formulated as a system of linear diophantine equations in 0-1 variables.

In our study we use the algorithm of Aardal, Hurkens, and Lenstra (1998) based on lattice basis reduction. This algorithm is not restricted to deal with market split instances only but is a general method for solving systems of linear diophantine equations with bounds on the variables.

We show computational results from solving both feasibility and optimization versions of the market split instances with up to 7 equations and 60 variables, and discuss various branching strategies and their effect on the number of nodes enumerated. To our knowledge, the largest feasibility and optimization instances solved before have 6 equations and 50 variables, and 4 equations and 30 variables respectively.

* Research partially supported by the ESPRIT Long Term Research Project nr. 20244 (Project ALCOM-IT: *Algorithms and Complexity in Information Technology*), by the project TMR-DONET nr. ERB FMRX-CT98-0202, both of the European Community, and by NSF through the Center for Research on Parallel Computation, Rice University, under Cooperative Agreement No. CCR-9120008.

** Research partially supported by the project TMR-DONET nr. ERB FMRX-CT98-0202 of the European Community.

*** Research partially supported by the ESPRIT Long Term Research Project nr. 20244 (Project ALCOM-IT: *Algorithms and Complexity in Information Technology*), and by the project TMR-DONET nr. ERB FMRX-CT98-0202, both of the European Community.

We also present a probabilistic analysis describing how to compute the probability of generating infeasible market split instances. The formula used by Cornuéjols and Dawande tends to produce relatively many feasible instances for sizes larger than 5 equations and 40 variables.

1 Introduction and Problem Description

The feasibility version of the market split problem is described as follows. A company with two divisions supplies retailers with several products. The goal is to allocate each retailer to one of the divisions such that division 1 controls $100c_i\%$, $0 \leq c_i \leq 1$, of the market for product i , and division 2 controls $(100 - 100c_i)\%$. There are n retailers and $m \leq n$ products. Let a_{ij} be the demand of retailer j for product i , and let d_i be determined as $\lfloor c_i d'_i \rfloor$, where d'_i is the total amount of product i that is supplied to the retailers. The decision variable x_j takes value 1 if retailer j is allocated to division 1 and 0 otherwise. The question is: “does there exist an allocation of the retailers to the divisions such that the desired market split is obtained?” One can formulate this problem mathematically as follows:

$$\mathbf{FP}: \text{ does there exist a vector } \mathbf{x} \in \mathbb{Z}^n : \mathbf{Ax} = \mathbf{d}, \quad \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}? \quad (1)$$

Let $X = \{\mathbf{x} \in \{0, 1\}^n : \sum_{j=1}^n a_{ij}x_j = d_i, \quad 1 \leq i \leq m\}$. Problem FP is NP-hard due to the bounds on the variables. The algorithm that we use was developed for the more general problem:

$$\text{does there exist a vector } \mathbf{x} \in \mathbb{Z}^n : \mathbf{Ax} = \mathbf{d}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}? \quad (2)$$

We assume that $\mathbf{A}$ is an integral $m \times n$ matrix, where $m \leq n$, $\mathbf{d}$ is an integral m -vector, and $\mathbf{l}$ and $\mathbf{u}$ are integral n -vectors. We denote the i th row of the matrix $\mathbf{A}$ by $\mathbf{a}_i$. Without loss of generality we assume that $\gcd(a_{i1}, a_{i2}, \dots, a_{in}) = 1$ for $1 \leq i \leq m$, and that $\mathbf{A}$ has full row rank.

In the optimization version of the market split problem we want to find the minimum slack, positive or negative, that needs to be added to the diophantine equations in order to make the system feasible:

$$\mathbf{OPT}: \min \left\{ \sum_{i=1}^m (s_i + w_i) \quad \text{s.t.} \quad (\mathbf{x}, \mathbf{s}, \mathbf{w}) \in X^S \right\}, \quad (3)$$

where $X^S = \{(\mathbf{x}, \mathbf{s}, \mathbf{w}) : \sum_{j=1}^n a_{ij}x_j + s_i - w_i = d_i, \quad 1 \leq i \leq m, \quad \mathbf{x} \in \{0, 1\}^n, \quad \mathbf{s}, \mathbf{w} \in \mathbb{Z}_+^m\}$.

The Cornuéjols-Dawande instances [2] of the market split problem were generated such that they were hard for linear programming based branch-and-bound, and they appear to be hard for several other methods as well. The input was generated as follows. Let $n = 10(m - 1)$ and let the coefficients a_{ij} be integer numbers drawn uniformly and independently from the interval $[0, D - 1]$, where $D = 100$. The right-hand-side coefficients are computed as

$d_i = \lfloor \frac{1}{2} \sum_{j=1}^n a_{ij} \rfloor$, $1 \leq i \leq m$. This corresponds to a market split where $c_i = \frac{1}{2}$ for $1 \leq i \leq m$. Cornuéjols and Dawande [2] argued that with this choice of data most of the instances of the feasibility problem (1) are infeasible, which implies that the optimization variant (3) has an objective value greater than zero. If branch-and-bound is used to solve OPT (3), then, due to the symmetry of the input, the value of the LP-relaxation remains at zero even after many variables have been fixed. A behaviour of branch-and-bound that was observed by Cornuéjols and Dawande was that $2^{\alpha n}$ nodes needed to be evaluated, where α typically takes values between 0.6 and 0.7.

The algorithm we use in our study is described briefly in Section 2. The algorithm was developed by Aardal, Hurkens, and Lenstra [1] for solving a system of linear diophantine equations with bounds on the variables, such as problem (2), and is based on Lovász' lattice basis reduction algorithm as described by Lenstra, Lenstra, and Lovász [6]. Aardal et al. motivate their choice of basis reduction as the main ingredient of their algorithm by arguing that one can interpret problem (2) as checking whether there exists a *short* integral vector satisfying the system $\mathbf{A}\mathbf{x} = \mathbf{d}$. Given the lattice, the basis reduction algorithm finds a basis spanning that lattice such that the basis consists of short, nearly orthogonal vectors. Hence, a lattice is chosen that seems particularly useful for problem (2). An initial basis that spans the given lattice is derived, and the basis reduction algorithm is applied to this basis. The parameters of the initial basis are chosen such that the reduced basis contains one vector $\mathbf{x}_d$ satisfying $\mathbf{A}\mathbf{x}_d = \mathbf{d}$, and $n - m$ linearly independent vectors $\mathbf{x}_0$ satisfying $\mathbf{A}\mathbf{x}_0 = \mathbf{0}$. Due to the basis reduction algorithm all these vectors are relatively short. If the vector $\mathbf{x}_d$ satisfies the bounds, then the algorithm terminates, and if not, one observes that $\mathbf{A}(\mathbf{x}_d + \lambda\mathbf{x}_0) = \mathbf{d}$ for any integer multiplier λ and any vector $\mathbf{x}_0$ such that $\mathbf{A}\mathbf{x}_0 = \mathbf{0}$. Hence, one can branch on integer linear combinations of vectors $\mathbf{x}_0$ satisfying $\mathbf{A}\mathbf{x}_0 = \mathbf{0}$ in order to obtain either a vector satisfying the diophantine equations as well as the lower and upper bounds, or a proof that no such vector exists.

In our computational study we solve both feasibility and optimization versions of the market split problem. The optimization version can be solved by a slightly adapted version of the Aardal-Hurkens-Lenstra algorithm. We have solved instances with up to 7 equations and 60 variables. To our knowledge, the largest feasibility instances solved so far had 6 constraints and 50 variables, and the largest optimization instances had 4 constraints and 30 variables. These results were reported by Cornuéjols and Dawande [2]. Our computational experience is presented in Section 3.

When performing the computational study we observed that the larger the instances became, the more often feasible instances were generated. This motivated us to analyse the expected number of solutions for instances generated according to Cornuéjols and Dawande. Our conclusion is that for a given value of $m > 4$, one needs to generate slightly less variables than is given by the expression $n = 10(m - 1)$ (keeping all other parameters the same). We present our analysis together with numerical support in Section 4.

2 An Outline of the Algorithm

Here we give a summary of the algorithm developed by Aardal, Hurkens, and Lenstra [1] to solve problem (2). They also give a brief overview of the basis reduction algorithm and the use of basis reduction in integer programming. For a detailed description of the basis reduction algorithm we refer to Lenstra, Lenstra, and Lovász [6].

The main idea behind the algorithm is to use an integer relaxation of the set $X = \{x \in \mathbb{Z}^n : \mathbf{A}x = \mathbf{d}, \mathbf{l} \leq x \leq \mathbf{u}\}$. The relaxation that Aardal et al. consider is $X_{IR} = \{x \in \mathbb{Z}^n : \mathbf{A}x = \mathbf{d}\}$. To determine whether X_{IR} is empty can be done in polynomial time. Aardal et al. rewrite the set X_{IR} as follows:

$$X_{IR} = \{x \in \mathbb{Z}^n : x = x_d + \mathbf{X}_0\lambda, \lambda \in \mathbb{Z}^{n-m}\}, \quad (4)$$

where $x_d \in \mathbb{Z}^n$ satisfies $\mathbf{A}x_d = \mathbf{d}$, and where $\mathbf{X}_0$ is an integral $n \times (n - m)$ matrix such that the columns x_0^j , $1 \leq j \leq n - m$, of $\mathbf{X}_0$ are linearly independent and such that each column x_0^j satisfies $\mathbf{A}x_0^j = \mathbf{0}$. Note that reformulation (4) is not unique. Expression (4) states that any integer vector x satisfying $\mathbf{A}x = \mathbf{d}$ can be described as a vector x_d , satisfying $\mathbf{A}x_d = \mathbf{d}$, plus an integer linear combination of vectors that form a basis for the lattice $\{x \in \mathbb{Z}^n : \mathbf{A}x = \mathbf{0}\}$. Of course this observation also holds for the integer vectors x satisfying the bound constraints, if such vectors exist. Aardal et al. argue that if we are able to find a vector x_d that is reasonably short, then this vector will hopefully satisfy the bound constraints. If that is the case, one is done, and if not one needs to check whether there exists an integer linear combination of the columns of $\mathbf{X}_0$, $\mathbf{X}_0\lambda$, such that $x = x_d + \mathbf{X}_0\lambda$ satisfies the bounds.

One way of obtaining the vectors x_d and x_0^j , $1 \leq j \leq n - m$, is by using the Hermite normal form of the matrix $\mathbf{A}$, see Schrijver [11], and Aardal et al. [1]. The numbers of the Hermite normal form, however, tend to be relatively large, whereas we want the vector x_d to contain small elements. Moreover having the other numbers in the computation large may cause numerical problems depending on which branching strategy we apply. Aardal et al. therefore chose to use lattice basis reduction to derive the vectors x_d and x_0^j , $1 \leq j \leq n - m$.

Let $\mathbf{I}^{(p)}$ denote the p -dimensional identity matrix, and let $\mathbf{0}^{(p \times q)}$ denote the $(p \times q)$ -matrix consisting of only zeros. Let N_1 and N_2 be positive integral numbers. Consider the following linearly independent column vectors $\mathbf{B} = (b_j)_{1 \leq j \leq n+1}$:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}^{(n)} & \mathbf{0}^{(n \times 1)} \\ \mathbf{0}^{(1 \times n)} & N_1 \\ N_2 \mathbf{A} & -N_2 \mathbf{d} \end{pmatrix}. \quad (5)$$

The vectors of $\mathbf{B}$ span the lattice $L \subset \mathbb{R}^{n+m+1}$ that contains vectors of the form

$$(x^T, N_1 y, N_2(a_1 x - d_1 y), \dots, N_2(a_m x - d_m y))^T = \mathbf{B} \begin{pmatrix} x \\ y \end{pmatrix}, \quad (6)$$

where y is a variable associated with the right-hand-side vector $\mathbf{d}$.

Proposition 1. *The integer vector $\mathbf{x}_d$ satisfies $\mathbf{A}\mathbf{x}_d = \mathbf{d}$ if and only if the vector*

$$(\mathbf{x}_d^T, N_1, \mathbf{0}^{(1 \times m)})^T = \mathbf{B} \begin{pmatrix} \mathbf{x}_d \\ 1 \end{pmatrix} \quad (7)$$

belongs to the lattice L , and the integer vector $\mathbf{x}_0$ satisfies $\mathbf{A}\mathbf{x}_0 = \mathbf{0}$ if and only if the vector

$$(\mathbf{x}_0^T, 0, \mathbf{0}^{(1 \times m)})^T = \mathbf{B} \begin{pmatrix} \mathbf{x}_0 \\ 0 \end{pmatrix} \quad (8)$$

belongs to the lattice L .

Aardal, Hurkens, and Lenstra [1] proved that if there exists an integer vector $\mathbf{x}$ satisfying the system $\mathbf{A}\mathbf{x} = \mathbf{d}$, and if the numbers N_1 and N_2 are chosen appropriately, i.e., large enough with respect to the input and relative to each other, then the first $n - m + 1$ columns of the reduced basis that is obtained by applying the basis reduction algorithm to $\mathbf{B}$ will be of the following form:

$$\begin{pmatrix} \mathbf{X}_0^{(n \times (n-m))} & \mathbf{x}_d \\ \mathbf{0}^{(1 \times (n-m))} & N_1 \\ \mathbf{0}^{(m \times (n-m))} & 0 \end{pmatrix}, \quad (9)$$

where $\mathbf{X}_0 = (\mathbf{x}_0^1, \dots, \mathbf{x}_0^{(n-m)})$. Due to Proposition 1 we can conclude that $\mathbf{x}_d$ satisfies $\mathbf{A}\mathbf{x}_d = \mathbf{d}$, and that $\mathbf{A}\mathbf{x}_0^j = \mathbf{0}$, $1 \leq j \leq n - m$.

Problem (2) can now be formulated equivalently as (cf. (4)):

$$\text{does there exist a vector } \boldsymbol{\lambda} \in \mathbb{Z}^{(n-m)} \text{ s. t. } \mathbf{l} - \mathbf{x}_d \leq \mathbf{X}_0 \boldsymbol{\lambda} \leq \mathbf{u} - \mathbf{x}_d? \quad (10)$$

Unless $\mathbf{x}_d$ satisfies the bound constraints, one needs to branch on the variables λ_j in order to check whether the polytope $P = \{\boldsymbol{\lambda} \in \mathbb{Z}^{(n-m)} : \mathbf{l} - \mathbf{x}_d \leq \mathbf{X}_0 \boldsymbol{\lambda} \leq \mathbf{u} - \mathbf{x}_d\}$ is empty. The basis reduction algorithm runs in polynomial time. If one wants an overall algorithm that runs in polynomial time for a fixed number of variables, one needs to apply the algorithms of H.W. Lenstra, Jr. [7] or of Lovász and Scarf [9]. Otherwise, one can, for instance, apply integral branching on the unit vectors in $\boldsymbol{\lambda}$ -space or linear programming based branch-and-bound. By integral branching in the $\boldsymbol{\lambda}$ -space we mean the following. Assume we are at node k of the search tree. Take any unit vector $\mathbf{e}_j$, i.e., the j th vector of the $(n - m)$ -dimensional identity matrix, that has not yet been considered at the predecessors of k . Measure the width of the polytope P in this direction, i.e., determine $u_k = \max\{\mathbf{e}_j^T \boldsymbol{\lambda} : \boldsymbol{\lambda} \in P \cap \{\lambda_j \text{'s fixed at predecessors of } k\}\}$ and $l_k = \min\{\mathbf{e}_j^T \boldsymbol{\lambda} : \boldsymbol{\lambda} \in P \cap \{\lambda_j \text{'s fixed at predecessors of } k\}\}$. Create $\lfloor u_k \rfloor - \lceil l_k \rceil + 1$ subproblems at node k of the search tree by fixing λ_j to the values $\lceil l_k \rceil, \dots, \lfloor u_k \rfloor$. The question is now in which order we should choose the unit vectors in our branching scheme. One alternative is to just take them in any predetermined order, and another is to determine, at node k , which unit vector yields the smallest value of $\lfloor u_k \rfloor - \lceil l_k \rceil$. This branching scheme is similar to the scheme proposed by Lovász and Scarf [9]

except that we in general are not sure whether the number of branches created at each level is bounded by some constant depending only on the dimension. What we hope for, and what seems to be the case given our computational results, is that $\lfloor u_k \rfloor - \lceil l_k \rceil$ is small at most nodes of the tree. A natural question in the context of branching is whether we may hope that linear programming based branch-and-bound is more efficient on the polytope P as compared to the polytope X . As can be observed in Section 3 we typically reduce the number of nodes if we branch on P instead of X by several orders of magnitude. One way of explaining this is that we obtain a scaling effect by going from description (2) of our problem to description (10), see Aardal et al. [1].

Table 1. Results for the feasibility version

Inst.	m	n	type	LP B&B on λ		thinnest e_j		$e_j, j = (n - m), \dots, 1$	
				# nodes	time (s)	# nodes	time (s)	# nodes	time (s)
M1	5	40	N	7,203	40	1,723	652	3,107	58
M2	5	40	N	10,488	58	3,839	1,335	8,252	118
M3	5	40	Y	5,808	30	1,398	558	2,155	39
M4	5	40	N	16,484	84	5,893	1,685	14,100	175
M5	5	40	N	17,182	94	3,027	1,163	5,376	85
M6	5	40	N	11,500	62	2,762	1,010	5,322	87
M7	5	40	N	16,666	88	4,025	1,391	9,710	123
M8	5	40	N	7,483	42	2,386	899	4,310	68
M9	5	40	N	6,393	36	1,674	660	3,115	50
M10	5	40	N	17,206	90	3,791	1,319	7,860	115
M11	6	50	N	413,386	3,690	88,619	53,713	152,399	3,532
M12	6	50	Y	72,093	516	14,456	9,078	8,479	229
M13	6	50	N	375,654	3,080	86,505	51,626	141,259	3,596
M14	6	50	N	381,813	2,984	108,725	75,794	204,367	4,193
M15	6	50	Y	125,250	960	69,981	37,774	129,402	2,700
M16	6	50	N	114,215	932	40,274	24,969	79,130	1,639
M17	7	60	N	108,154	1,228	36,288	36,023	73,877	2,095

3 Computational Experience

3.1 The Feasibility Version

We solved 17 instances of problem FP (1) reformulated as problem (10). Three of the instances were feasible and 14 infeasible. The input was generated as described in Section 1. The instances M16 and M17 are the instances “markshare1” and “markshare2” of MIPLIB [10].

In our computational study we wanted to determine the effect of the reformulation of problem (1) to problem (10). Therefore, we solved formulation (10) by linear programming based branch-and-bound as it was used by Cornuéjols and Dawande [2] on formulation (1). Given that we consider an integer relaxation (4) rather than a linear relaxation when determining (10), we also wanted to investigate the effect of maintaining the integral representation by applying integral branching. This explains the choice of the three branching strategies that we considered in our study: linear programming based branch-and-bound on the variables λ_j , branching on the unit vector in λ -space that yields the smallest value of $\lfloor u_k \rfloor - \lceil l_k \rceil$ at node k of the search tree as described in Section 2, and branching on the unit vectors in the predetermined order $j = n - m, \dots, 1$, see Section 2. An example of an enumeration tree derived by the third strategy for an instance of size 4×30 is shown in Figure 1. Notice that few branches are created at each node of the tree, and that no branching occurs below level eight.

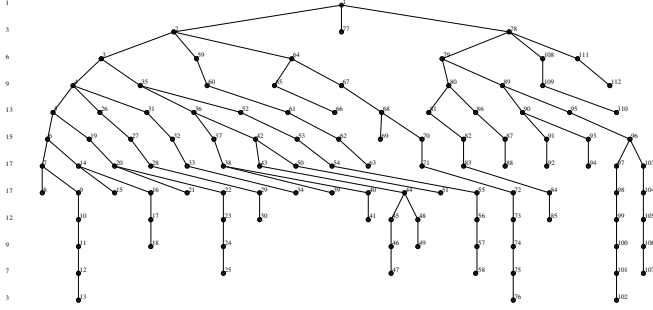


Fig. 1. Enumeration tree for integer branching e_j , $j = (n - m), \dots, 1$

The information in Table 1 is interpreted as follows. In the first three columns, “Instance”, “ m ”, and “ n ”, the instance names and the dimension of the instances are given. A “Y” in column “type” means that the instance is feasible, and an “N” that it is not feasible. Note that it is not known a priori whether the instances are feasible, but it is established by our algorithm. In the following six columns the number of nodes and the computing times are given for the three branching strategies. The computing times for the strategies “thinnest e_j ” and “ e_j , $j = (n - m), \dots, 1$ ”, are given in seconds on a Sun Ultra Enterprise 2 with two 168 MHz Ultra Sparc processors (our implementation is sequential), SpecInt95 6.34, SpecFp95 9.33. The linear programming branch-and-bound computations were carried out on an Alphaserber 4100 5/400 with four 400 MHz 21164 Alpha Processors (sequential implementation) SpecInt95 12.1, SpecFp95

17.2. The times reported on for this strategy are the actual times obtained with the Alphaserver multiplied by a factor of 2, in order to make it easier to compare the times of the three strategies. The basis reduction in the algorithm by Aardal, Hurkens, and Lenstra is done using LiDIA, a library for computational number theory [8]. The average computing time for the Aardal-Hurkens-Lenstra algorithm was for the three instance sizes 1.6, 3.1, and 4.8 seconds respectively. These computations were all carried out on the Sun Ultra Enterprise 2. For the LP-based branch-and-bound on the variables λ_j we used CPLEX 6.5 [4], and for the other two strategies we use the enumeration scheme developed by Verweij [12]. The linear programming subproblems that arise in these strategies when determining $\lfloor u_k \rfloor$ and $\lceil l_k \rceil$, are solved using CPLEX version 6.0.1 [3].

An important conclusion of the experiments is that the reformulation itself is essential. Cornuéjols and Dawande [2] could only solve instances up to size 4×30 using CPLEX version 4.0.3. We also applied CPLEX versions 6.0 and 6.5 to the initial problem formulation (1) and observed a similar behaviour, whereas CPLEX produced very good results on the reformulated problem (10). Cornuéjols and Dawande did solve feasibility instances of size 6×50 by a group relaxation approach. Their computing times are a factor of 3-10 slower than ours. No previous computational results for instances of size 7×60 have, to our knowledge, been reported.

If we consider the number of nodes that we enumerate in the three strategies, we note that branching on the thinnest unit vector (in λ -space) yields the fewest number of nodes for all instances except instance M12, which is a feasible instance. We do, however, need to determine the unit vector that yields the thinnest direction at each node of the tree, which explains the longer computing times. Branching on unit vectors in the predetermined order $j = n - m, \dots, 1$, also requires fewer nodes for most instances than the linear programming based branching on the variables λ_j . In terms of computing times, linear programming based branch-and-bound is for most instances the fastest strategy, but does not differ too much from the times needed for branching on unit vectors e_j , $j = n - m, \dots, 1$. This indicates that integer branching is an interesting strategy, in particular if we can find reasonably good branching directions quickly, as in the third strategy. In our case it seems as if the unit vectors in λ -space yield thin branching directions. To investigate this we applied the generalized basis reduction algorithm of Lovász and Scarf [9] to our polytope P . The reduced basis vectors yielded thinner directions than the strategy “thinnest e_j ” in only about 6% of the cases for the instances of size 5×40 . This implies that the unit vectors in λ -space, in some order, basically form a reduced basis in the Lovász-Scarf sense. The computations involved in determining a Lovász-Scarf reduced basis are fairly time consuming. For a problem of dimension 5×40 , at the root node of the tree, one has to solve at least 100 linear programs to determine the basis. For each level of the tree the number of linear programs solved at each node will decrease as the dimension of the subproblems decrease. If the unit basis would generate bad search directions, then a heuristic version of the Lovász-Scarf algorithm would be a possibility.

3.2 The Optimization Version

The algorithm by Aardal, Hurkens, and Lenstra [1] was primarily designed to solve feasibility problems, but can with simple adaptations be used to solve the optimization version (3) of the market split instances as well. Below, we report on the results obtained by using three different strategies to solve the optimization version. All strategies are based on linear programming based branch-and-bound.

Strategy 1: Here, we solve sequence of feasibility problems. We start with the feasibility version (10). If the instance is infeasible, then we proceed by considering the following sequence of feasibility problems for $v = 1, 2, \dots$ until a feasible solution is found.

$$\text{do there exist vectors } \mathbf{x} \in \{0, 1\}^n, \mathbf{s}, \mathbf{w} \in \mathbb{Z}_{\geq 0}^m : \quad (11)$$

$$\sum_{j=1}^n a_{ij}x_j + s_i - w_i = d_i, \quad 1 \leq i \leq m, \quad \sum_{i=1}^m (s_i + w_i) = v?$$

These feasibility problems are then reformulated as problems of type (10) using the algorithm of Aardal et al. For each of these feasibility problems we apply linear programming based branch-and-bound on the variables λ_j . Here, we also investigate the influence of the choice of objective function on the search that CPLEX is performing. In Strategy 1 we use the objective function $\max \sum_{j=1}^{n-m} \lambda_j$.

Strategy 2: This is the same as Strategy 1 except that the objective function is a perturbation of the objective function zero. Here we sketch the principle of the perturbation. What is basically done to construct the perturbed objective function is to perturb the variables of the linear programming dual as follows. Notice that the number of constraints in the linear relaxation of the reformulation (10) of the feasibility problem (11) is $p = 2n + 2m$; we have $2n$ constraints corresponding to the upper and lower bounds on the $\mathbf{x}$ -variables, and $2m$ constraints corresponding to the nonnegativity requirements on the slack variables s_i and w_i , $1 \leq i \leq m$. Let $\varepsilon = 10^{-6}$ and let, for $i = 1, \dots, p$, Z_i be a drawn uniformly and independently from the interval $[0, 1]$. Let $\delta_i = \varepsilon Z_i$. If the dual variable $y_i \leq 0$ in the original formulation we let $y_i \leq \delta_i$, and if $y_i \geq 0$ we let $y_i \geq -\delta_i$. For y_i such that $y_i \leq \delta_i$, make the substitution $Y_i = y_i - \delta_i$, and for $y_i \geq -\delta_i$ we substitute y_i by $Y_i = y_i + \delta_i$. This substitution implies a perturbation of the primal objective function.

Strategy 3: Here we view the problem as an optimization problem directly, which implies that only one problem is solved instead of a sequence of problems as in Strategies 1 and 2. We extract the expressions of the slack variables s_i and w_i in terms of the variables λ_j and minimize the sum of the slack variables expressed in the λ_j 's.

For all computations we used CPLEX version 6.5. The computations were made on an Alphaserver 4100 5/400 as described in the previous subsection.

Table 2. Results for the optimization version

Inst.	Strat. 1		Strat. 2		Strat. 3	
	# nodes	time (s)	# nodes	time (s)	# nodes	time (s)
M1	20,022	57	13,699	38	53,267	180
M2	6,451	44	75,174	222	109,498	281
M3	5,484	15	5,230	15	41,518	118
M4	36,847	107	75,985	211	176,437	456
M5	32,880	93	16,379	45	271,208	705
M6	35,710	105	21,277	62	220,603	875
M7	64,090	180	88,160	254	396,416	1,261
M8	33,937	99	35,471	103	122,526	383
M9	9,910	29	19,083	56	200,987	625
M10	28,402	76	146,224	436	99,502	299
M11	1,165,498	6,018	1,728,646	7,811	6,158,407	29,486
M12	53,273	214	73,801	283	5,101,843	22,002
M13	810,496	505	1,410,840	6,607	6,057,528	26,333
M14	384,882	1,505	1,117,107	4,748	7,861,402	34,083
M15	96,470	384	17,007	60	9,558,521	37,960
M16	282,665	1,190	319,029	1,248	71,253	348
M17	567,837	3,767	1,823,915	11,597	342,5941	27,168

From the results in Table 2 we can conclude that instances of sizes up to 7×60 are relatively easy to solve to optimality after using the reformulation of the problem implied by the algorithm of Aardal, Hurkens, and Lenstra [1]. This represents a large improvement over earlier results, where the largest optimization instances had dimension 4×30 , see Cornuéjols and Dawande [2]. If we consider the number of nodes that we enumerate when applying linear programming based branch-and-bound on the variables λ_j , we observe that this number is significantly smaller than the number $2^{\alpha n}$ for α between 0.6 and 0.7 that Cornuéjols and Dawande observed when applying branch-and-bound on the x_j -variables. For instances of size 4×30 they enumerated between 10^6 and 2×10^6 nodes. For the same number of enumeration nodes we can solve instances of more than twice that size. We can also observe that solving the reformulated optimization version (Strategy 3) instead of a sequence of feasibility problems (Strategies 1 and 2) is more time consuming in most cases. One reason is that the optimum objective value is small, so the number of problems we need to solve in Strategies 1 and 2 is small, but in case of the infeasible instances greater than one. If one expects the optimum value to be large and no reasonable bounds are known, then it is probably better to consider Strategy 3.

Next to the instances we report on here we also generated another five instances of size 7×60 . All these instances were feasible so we decided not to

report on the results of the computations here. For the size 6×50 we also had to generate quite a few instances to obtain infeasible ones. This motivated us to investigate whether the relation $n = 10(m - 1)$, as Cornuéjols and Dawande suggested, is actually likely to produce infeasible instances if we keep all other parameters as they suggested. Our probabilistic analysis is presented in the next section.

4 The Expected Number of Solutions

Here we derive an expression for the expected number of solutions for problem formulation (1), given that the coefficients a_{ij} are generated uniformly and independently from the set $\{0, \dots, D - 1\}$, and that $d_i = \lfloor p \sum_{j=1}^n a_{ij} \rfloor$, cf. Section 1. Cornuéjols and Dawande [2] use $D = 100$ and $p = \frac{1}{2}$.

4.1 The Probability that a Subset Induces a Solution

Consider a subset $S \subseteq \{1, 2, \dots, n\}$, and let $x_j = 1$ if $j \in S$, and $x_j = 0$ otherwise. We compute the probability that $\sum_{j \in S} a_{ij} = \lfloor p \sum_{j=1}^n a_{ij} \rfloor$, in which case the vector $\mathbf{x}$ as defined above satisfies $\mathbf{Ax} = \mathbf{d}$ for row i . Define a random variable $Z_i(S) = \sum_{j \in S} a_{ij} - \lfloor p \sum_{j=1}^n a_{ij} \rfloor$ denoting the difference between the left-hand side and the right-hand side of row i . The probability that we want to compute is therefore $\Pr[Z_i(S) = 0]$. Let random variables $Y_i(S)$ and U_i be defined as

$$Y_i(S) = \sum_{j \in S} a_{ij} - p \sum_{j=1}^n a_{ij} = \sum_{j \in S} (1 - p) a_{ij} - \sum_{j \notin S} p a_{ij}, \quad (12)$$

and $U_i = p \sum_{j=1}^n a_{ij} - \lfloor p \sum_{j=1}^n a_{ij} \rfloor$. Hence, we can write $Z_i(S) = Y_i(S) + U_i$.

For any rational fraction $p = P/Q$ ($P, Q \in \mathbb{N}$, $\gcd(P, Q) = 1$), we have $Y_i(S) \in \frac{1}{Q}\mathbb{Z}$ and $U_i \in \frac{1}{Q}\mathbb{Z} \cap [0, 1)$. Since $Y_i(S) + U_i \equiv 0 \pmod{1}$, we have

$$\Pr[Z_i(S) = 0] = \Pr[Y_i(S) = -U_i] = \sum_{k=0}^{Q-1} \Pr[Y_i(S) = \frac{-k}{Q}]. \quad (13)$$

We can compute this probability exactly using the probability generating function of $Y_i(S)$, see Section 4.2, or give an approximation using the normal distribution as described in Section 4.3. In either case, we obtain an expression $\Pr[Z_i(S) = 0] = q(n, D, |S|)$, i.e., the probability that $\mathbf{x}$ induced by S defines a solution for $(\mathbf{Ax})_i = \mathbf{d}_i$ depends on n , D and the size of S only. The probability that S induces a solution for $\mathbf{Ax} = \mathbf{d}$ is given by $q(n, D, |S|)^m$. The expected number of solutions is derived by summing over all subsets S , i.e.,

$$\mathbb{E}[\#\text{solutions}] = \sum_{S \subseteq \{1, \dots, n\}} q(n, D, |S|)^m = \sum_{s=0}^n \binom{n}{s} q(n, D, s)^m. \quad (14)$$

4.2 The Probability Generating Function

The probability generating function of a_{ij} is given by

$$G_{a_{ij}}(x) = \mathbb{E}[x^{a_{ij}}] = \sum_{i=0}^{D-1} \frac{1}{D} x^i = \frac{1}{D} (1 + x + \dots + x^{D-1}) = \frac{1}{D} \frac{x^D - 1}{x - 1}. \quad (15)$$

Similarly, the probability generating function of $Y_i(S)$ is given by

$$G_{Y_i(S)}(x) = \prod_{j \in S} \frac{1}{D} \frac{x^{(1-p)D} - 1}{x^{(1-p)} - 1} \prod_{j \notin S} \frac{1}{D} \frac{x^{(-p)D} - 1}{x^{(-p)} - 1} \quad (16)$$

$$= \left(\frac{1}{D}\right)^n \left(\frac{x^{(1-p)D} - 1}{x^{(1-p)} - 1}\right)^{|S|} \left(\frac{x^{(-p)D} - 1}{x^{(-p)} - 1}\right)^{n-|S|} \quad (17)$$

$$= \frac{1}{D^n} \frac{1}{x^{p(D-1)(n-|S|)}} \left(\frac{x^{(1-p)D} - 1}{x^{(1-p)} - 1}\right)^{|S|} \left(\frac{x^{pD} - 1}{x^p - 1}\right)^{n-|S|}. \quad (18)$$

For rational $p = P/Q$, we would like to expand expression (18) to $\sum_j c_j x^{j/Q}$, where c_j denotes the probability $\Pr[Y_i(S) = \frac{j}{Q}]$. To compute $\Pr[Z_i(S) = 0]$, we only need to evaluate the coefficients c_j for $j = 0, -1, \dots, -Q + 1$. For $p = \frac{1}{2}$, expression (18) is equal to

$$\frac{1}{D^n} \frac{1}{x^{(D-1)(n-|S|)/2}} \left(\frac{x^{D/2} - 1}{x^{1/2} - 1}\right)^n. \quad (19)$$

The major factors in (18) and (19) are of the form $(\frac{y^D - 1}{y - 1})^T$, which is equal to $\sum_{j=0}^{\infty} a_j y^j$ with

$$a_j = \sum_{k=0}^{\min\{T, \lfloor j/D \rfloor\}} \binom{T}{k} (-1)^k \binom{j - Dk + T - 1}{j - Dk}. \quad (20)$$

For $p = \frac{1}{2}$, we take $T = n$ and $y = x^{1/2}$, to obtain

$$\Pr[Z_i(S) = 0] = q(n, D, |S|) = \frac{1}{D^n} (a_{(D-1)(n-|S|)} + a_{(D-1)(n-|S|)-1}). \quad (21)$$

For $p \neq \frac{1}{2}$, we use (20) to compute the coefficients of each factor in expression (18) and derive the c_j -coefficients by convolution of the two power series obtained.

4.3 An Approximation Using the Normal Distribution

We can also approximate $q(n, D, |S|)$ using the normal distribution. Each of the coefficients a_{ij} has expectation $\frac{1}{2}(D-1)$ and variance $\frac{1}{12}(D^2-1)$. Since they are drawn independently, we obtain $\mathbb{E}[Y_i(S)] = \frac{1}{2}(D-1)(|S| - np)$ and $\text{Var}[Y_i(S)] =$

$\frac{1}{12}(D^2 - 1)(|S|(1 - 2p) + p^2n)$. Note that for $p = \frac{1}{2}$, the variance reduces to $(D^2 - 1)n/48$. For rational $p = P/Q$, the probability that subset S induces a solution for row i is given by $\Pr[Z_i(S) = 0] = \Pr[1/(2Q) - 1 \leq Y_i(S) \leq 1/(2Q)]$. We can approximate this expression, using the Central Limit Theorem [5], by the normal distribution giving

$$\Pr\left[\frac{1}{2Q} - 1 < Y_i(S) < \frac{1}{2Q}\right] \approx \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\beta} \exp\left(-\frac{1}{2}u^2\right) du, \quad (22)$$

with

$$\alpha = \frac{\frac{1}{2Q} - 1 - \mathbb{E}[Y_i(S)]}{\sqrt{\text{Var}[Y_i(S)]}}, \quad \text{and} \quad \beta = \frac{\frac{1}{2Q} - \mathbb{E}[Y_i(S)]}{\sqrt{\text{Var}[Y_i(S)]}}. \quad (23)$$

4.4 The Probability of Generating Infeasible Instances

Finally, we approximate the probability of drawing an infeasible instance. Here, we neglect the dependency between two distinct subsets, each not providing a solution. We further use $\log(1 + x) = x + O(x^2)$, finding

$$\Pr[\#\text{solutions} = 0] \approx \prod_{S \subset \{1, \dots, n\}} \Pr[S \text{ is no solution}] \quad (24)$$

$$= \exp\left(\log \prod_{S \subset \{1, \dots, n\}} (1 - q(n, D, |S|)^m)\right) \quad (25)$$

$$= \exp\left(\sum_{S \subset \{1, \dots, n\}} \log(1 - q(n, D, |S|)^m)\right) \quad (26)$$

$$\approx \exp\left(\sum_{S \subset \{1, \dots, n\}} -q(n, D, |S|)^m\right) \quad (27)$$

$$= \exp(-\mathbb{E}[\#\text{solutions}]) . \quad (28)$$

4.5 Computational Results

We have computed the expected number of solutions using probability generating functions (gen), and using an approximation by the normal distribution (approx) for several values of m and n . The results are presented in Table 3. In our computations we use $p = \frac{1}{2}$ and $D = 100$. The horizontal lines in the table indicate the relation $n = 10(m - 1)$ as proposed by Cornuéjols and Dawande. We notice that the value obtained by the approximation overestimates the exact value with a relative error of at most 5.2 %.

For a given value of m the table shows that the expected number of solutions, and therefore the probability of drawing a feasible instance, increases rapidly when n increases, see also Figure 2. In particular, we observe that for $m \geq 6$, the expected number of solutions using the relation $n = 10(m - 1)$ is

greater than 0.9. This confirms our experience with the instances we drew for our computational experiments reported on in Section 3. If one wants to generate infeasible instances for $m \geq 6$ with high probability, then one needs to generate slightly fewer columns for a given value of m than the relation $n = 10(m - 1)$ indicates.

Table 3. The expected number of solutions computed exactly using the probability generating function (gen) and approximated by the normal distribution (approx)

n	$m = 4$		$m = 5$		$m = 6$		$m = 7$		$m = 8$	
	gen	approx	gen	approx	gen	approx	gen	approx	gen	approx
20	0.0004	0.0004	2.3091e-6	2.3692e-6	1.3071e-8	1.3519e-8	7.5214e-11	7.8457e-11	4.3865e-13	4.6155e-13
21	0.0007	0.0008	4.0743e-6	4.1700e-6	2.2299e-8	2.2964e-8	1.2316e-10	1.2759e-10	6.8386e-13	7.1256e-13
22	0.0014	0.0014	7.2921e-6	7.4623e-6	3.9302e-8	4.0507e-8	2.1514e-10	2.2339e-10	1.1927e-12	1.2479e-12
23	0.0025	0.0026	0.00001	0.00001	6.8320e-8	7.0205e-8	3.6171e-10	3.7380e-10	1.9270e-12	2.0023e-12
24	0.0046	0.0047	0.00002	0.00002	1.2125e-7	1.2461e-7	6.3449e-10	6.5644e-10	3.3604e-12	3.5006e-12
25	0.0086	0.0087	0.00004	0.00004	2.1378e-7	2.1924e-7	1.0881e-9	1.1219e-9	5.5765e-12	5.7801e-12
26	0.0159	0.0161	0.00008	0.00008	3.8208e-7	3.9177e-7	1.9192e-9	1.9797e-9	9.7509e-12	1.0121e-11
27	0.0295	0.0298	0.0001	0.0001	6.8118e-7	6.9740e-7	3.3415e-9	3.4385e-9	1.6516e-11	1.7079e-11
28	0.0548	0.0555	0.0003	0.0003	1.2257e-6	1.2544e-6	5.9298e-9	6.1020e-9	2.9002e-11	3.0013e-11
29	0.1023	0.1035	0.0005	0.0005	2.2050e-6	2.2540e-6	1.0449e-8	1.0733e-8	4.9914e-11	5.1513e-11
30	0.1913	0.1936	0.0009	0.0009	3.9929e-6	4.0798e-6	1.8657e-8	1.9159e-8	8.8096e-11	9.0939e-11
31	0.3585	0.3626	0.0016	0.0016	7.2371e-6	7.3878e-6	3.3199e-8	3.4045e-8	1.5357e-10	1.5819e-10
32	0.6732	0.6808	0.0030	0.0030	0.00001	0.00001	5.9626e-8	6.1124e-8	2.7250e-10	2.8068e-10
33	1.2668	1.2805	0.0055	0.0055	0.00002	0.00002	1.0696e-7	1.0953e-7	4.8000e-10	4.9361e-10
34	2.3879	2.4131	0.0102	0.0103	0.00004	0.00004	1.9319e-7	1.9774e-7	8.5633e-10	8.8043e-10
35	4.5090	4.5552	0.0189	0.0192	0.00008	0.00008	3.4895e-7	3.5685e-7	1.5215e-9	1.5623e-9
36	8.5279	8.6128	0.0353	0.0358	0.0001	0.0001	6.3353e-7	6.4759e-7	2.7288e-9	2.8010e-9
37	16.1533	16.3098	0.0659	0.0668	0.0003	0.0003	1.1512e-6	1.1758e-6	4.8844e-9	5.0085e-9
38	30.6410	30.9297	0.1234	0.1250	0.0005	0.0005	2.1000e-6	2.1440e-6	8.8038e-9	9.0240e-9
39	58.2021	58.7363	0.2314	0.2343	0.0009	0.0009	3.8357e-6	3.9137e-6	1.5859e-8	1.6241e-8
40	110.6973	111.6878	0.4345	0.4400	0.0017	0.0018	7.0282e-6	7.1681e-6	2.8719e-8	2.9400e-8
41	210.7999	212.6397	0.8174	0.8274	0.0032	0.0033	0.00001	0.00001	5.2021e-8	5.3216e-8
42	401.8960	405.3196	1.5399	1.5582	0.0060	0.0061	0.00002	0.00002	9.4624e-8	9.6756e-8
43	767.0835	773.4649	2.9050	2.9388	0.0112	0.0114	0.00004	0.00004	1.7224e-7	1.7601e-7
44	1465.6670	1477.5812	5.4876	5.5499	0.0209	0.0212	0.00008	0.00008	3.1459e-7	3.2134e-7
45	2803.3082	2825.5860	10.3793	10.4945	0.0391	0.0397	0.0001	0.0002	5.5717e-7	5.7012e-7
46	5366.9806	5408.6988	19.6556	19.8690	0.0733	0.0743	0.0003	0.0003	1.0545e-6	1.0761e-6
47	1.0285e4	1.0363e4	37.2658	37.6617	0.1375	0.1393	0.0005	0.0005	1.9356e-6	1.9744e-6
48	1.9726e4	1.9874e4	70.7327	71.4684	0.2582	0.2617	0.0010	0.0010	3.5613e-6	3.6313e-6
49	3.7868e4	3.8145e4	134.3989	135.7680	0.4856	0.4920	0.0018	0.0018	6.0567e-6	6.6867e-6
50	7.2753e4	7.3275e4	255.6339	258.1855	0.9144	0.9262	0.0033	0.0034	0.00001	0.00001
51	1.3989e5	1.4087e5	486.7099	491.4721	1.7238	1.7457	0.0062	0.0062	0.00002	0.00002
52	2.6918e5	2.7103e5	927.5451	936.4449	3.2537	3.2940	0.0116	0.0117	0.00004	0.00004
53	5.1834e5	5.2184e5	1769.2812	1785.9346	6.1478	6.2225	0.0216	0.0220	0.00008	0.00008
54	9.9884e5	1.0054e6	3377.8566	3409.0581	11.6287	11.7673	0.0405	0.0411	0.0001	0.0001
55	1.9261e6	1.9386e6	6454.3702	6512.8979	22.0181	22.2758	0.0761	0.0772	0.0003	0.0003
56	3.7165e6	3.7402e6	1.2343e4	1.2453e4	41.7308	42.2104	0.1429	0.1449	0.0005	0.0005
57	7.1757e6	7.2207e6	2.3622e4	2.3830e4	79.1668	80.0606	0.2687	0.2724	0.0009	0.0009
58	1.3863e7	1.3948e7	4.5245e4	4.5635e4	150.3239	151.9916	0.5058	0.5127	0.0017	0.0017
59	2.6799e7	2.6961e7	8.6723e4	8.7457e4	285.6905	288.8057	0.9531	0.9658	0.0032	0.0033
60	5.1835e7	5.2143e7	1.6634e5	1.6772e5	543.4188	549.2448	1.7978	1.8214	0.0060	0.0061
61	1.0031e8	1.0090e8	3.1928e5	3.2189e5	1034.5026	1045.4103	3.3943	3.4383	0.0112	0.0114
62	1.9424e8	1.9535e8	6.1324e5	6.1817e5	1970.9501	1991.3942	6.4148	6.4966	0.0211	0.0214
63	3.7629e8	3.7843e8	1.1786e6	1.1879e6	3757.9874	3796.3444	12.1341	12.2862	0.0395	0.0401
64	7.2936e8	7.3343e8	2.2666e6	2.2842e6	7170.6817	7242.7199	22.9726	23.2561	0.0733	0.0754
65	1.4144e9	1.4221e9	4.3616e6	4.3951e6	1.3692e4	1.3828e4	43.5290	44.0578	0.1397	0.1417
66	2.7440e9	2.7589e9	8.3980e6	8.4614e6	2.6164e4	2.6419e4	82.5476	83.5352	0.2629	0.2666
67	5.3262e9	5.3545e9	1.6179e7	1.6299e7	5.0029e4	5.0511e4	156.6664	158.5126	0.4952	0.5021
68	1.0343e10	1.0396e10	3.1186e7	3.1412e7	9.5728e4	9.6634e4	297.5662	301.0208	0.9337	0.9465
69	2.0092e10	2.0196e10	6.0146e7	6.0581e7	1.8328e5	1.8499e5	565.6096	572.0801	1.7619	1.7858
70	3.9050e10	3.9248e10	1.1606e8	1.1688e8	3.5114e5	3.5437e5	1075.8876	1088.0185	3.3274	3.3720
71	7.5923e10	7.6305e10	2.2406e8	2.2563e8	6.7315e5	6.7925e5	2047.9738	2070.7375	6.2893	6.3723
72	1.4767e11	1.4840e11	4.3279e8	4.3578e8	1.2912e6	1.3027e6	3901.0477	3943.8021	11.8969	12.0517
73	2.8734e11	2.8875e11	8.3636e8	8.4207e8	2.4781e6	2.4990e6	7435.8186	7516.1883	22.5215	22.8106
74	5.5932e11	5.6202e11	1.6170e9	1.6278e9	4.7588e6	4.8001e6	1.4183e4	1.4334e4	42.6664	43.2066
75	1.0891e12	1.0943e12	3.1277e9	3.1484e9	9.1434e6	9.2218e6	2.7068e4	2.7354e4	80.8888	81.8993
76	2.1215e12	2.1314e12	6.0523e9	6.0920e9	1.7577e7	1.7725e7	5.1694e4	5.2251e4	153.4610	155.3527
77	4.1339e12	4.1531e12	1.1717e10	1.1792e10	3.3807e7	3.4089e7	9.7881e4	9.9795e4	291.3437	294.8880
78	8.0580e12	8.0948e12	2.2693e10	2.2837e10	6.5057e7	6.5593e7	1.8887e5	1.9078e5	553.4833	560.1292
79	1.5712e13	1.5783e13	4.3968e10	4.4245e10	1.2525e8	1.2627e8	3.6133e5	3.6494e5	1052.1712	1064.6447
80	3.0646e13	3.0782e13	8.5223e10	8.5754e10	2.4126e8	2.4319e8	6.9164e5	6.9847e5	2001.4513	2024.8798

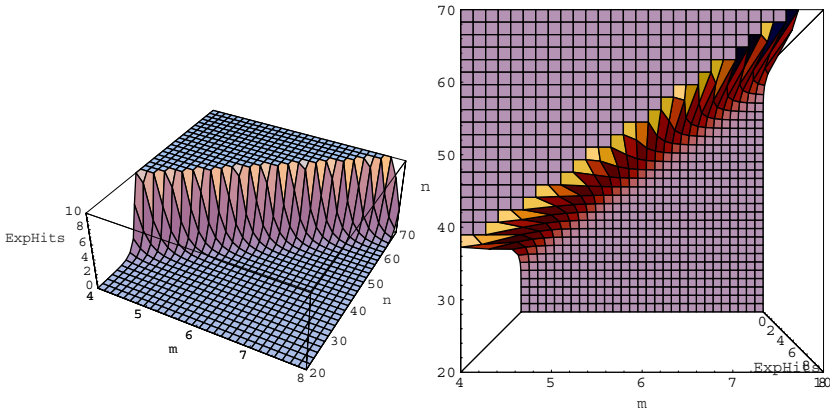


Fig. 2. The expected number of solutions for $m = 4, \dots, 8$, truncated at 10

Acknowledgements

We would like to thank Bram Verweij for his assistance in implementing our integral branching algorithm using his enumeration scheme [12]. We also want to thank David Applegate and Bill Cook for their many useful comments on our work and for allowing us to use their DEC Alphaservers.

References

1. K. Aardal, C. Hurkens, A. K. Lenstra (1998). Solving a system of diophantine equation with lower and upper bounds on the variables. Research report UU-CS-1998-36, Department of Computer Science, Utrecht University.
2. G. Cornuéjols, M. Dawande (1998). A class of hard small 0-1 programs. In: R. E. Bixby, E. A. Boyd, R. Z. Ríos-Mercado (eds.) *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference*. Lecture Notes in Computer Science 1412, pp 284–293, Springer-Verlag, Berlin Heidelberg.
3. *CPLEX 6.0 Documentation Supplement* (1998). ILOG Inc., CPLEX Division, Incline Village NV.
4. *CPLEX 6.5 Documentation Supplement* (1999). ILOG Inc., CPLEX Division, Incline Village NV.
5. G. R. Grimmett, D. R. Stirzaker (1982). *Probability and Random Processes*, Oxford University Press, Oxford.
6. A. K. Lenstra, H. W. Lenstra, Jr., L. Lovász (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen* 261, 515–534.
7. H. W. Lenstra, Jr. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research* 8, 538–548.
8. LiDIA – A library for computational number theory. TH Darmstadt / Universität des Saarlandes, Fachbereich Informatik, Institut für Theoretische Informatik.
<http://www.informatik.th-darmstadt.de/pub/TI/LiDIA>

9. L. Lovász, H. E. Scarf (1992). The generalized basis reduction algorithm. *Mathematics of Operations Research* 17, 751–764.
10. MIPLIB. <http://www.caam.rice.edu/~bixby/miplib/miplib3.html>
11. A. Schrijver (1986). *Theory of Integer and Linear Programming*. Wiley, Chichester.
12. A. M. Verweij (1998). *The UHFCO Library*. Department of Computer Science, Utrecht University.

Approximation Algorithms for Maximum Coverage and Max Cut with Given Sizes of Parts^{*}

Alexander A. Ageev and Maxim I. Sviridenko

Sobolev Institute of Mathematics
pr. Koptyuga 4, 630090, Novosibirsk, Russia
{ageev,svir}@math.nsc.ru

Abstract. In this paper we demonstrate a general method of designing constant-factor approximation algorithms for some discrete optimization problems with cardinality constraints. The core of the method is a simple deterministic (“pipage”) procedure of rounding of linear relaxations. By using the method we design a $(1 - (1 - 1/k)^k)$ -approximation algorithm for the maximum coverage problem where k is the maximum size of the subsets that are covered, and a $1/2$ -approximation algorithm for the maximum cut problem with given sizes of parts in the vertex set bipartition. The performance guarantee of the former improves on that of the well-known $(1 - e^{-1})$ -greedy algorithm due to Cornuejols, Fisher and Nemhauser in each case of bounded k . The latter is, to the best of our knowledge, the first constant-factor algorithm for that version of the maximum cut problem.

1 Introduction

It is a fact of the present day that rounding of linear relaxations is one of the most effective techniques in designing approximation algorithms with proven worst-case bounds for discrete optimization problems. The quality characteristics of a rounding-based algorithm are highly dependent on the choice of an integer program reformulation and a rounding scheme. When applying popular random roundings one encounters the additional problem of derandomization. This problem may prove to be extremely difficult or quite intractable. For example, the widely known derandomization method of conditional probabilities [1] succeeds, as is easily seen, only under some very special conditions; in particular, if the relaxation has a subset of active variables that determine the optimal values of the remaining ones and the optimization problem with respect to these active variables is unconstrained. If one adds even a single cardinality constraint connecting the active variables, the method fails. In this paper we present a simple deterministic (“pipage”) rounding method to tackle some problems of this sort. So far the method has happened to be applicable to two well-known

^{*} This research was partially supported by the Russian Foundation for Basic Research, grant 97-01-00890

problems. By using it we design a $(1 - (1 - 1/k)^k)$ -approximation algorithm for the maximum coverage problem where k is the maximum size of the subsets, and a $1/2$ -approximation algorithm for the maximum cut problem with given sizes of parts in the vertex set bipartition. The performance guarantee of the former improves on that of the well-known $(1 - e^{-1})$ -greedy algorithm due to Cornuejols, Fisher and Nemhauser [2] in each case of bounded k . The latter is, to the best of our knowledge, the first constant-factor algorithm for that version of the maximum cut problem. In Sections 4 and 5 we show that both algorithms can be extended to apply to more general problems — the maximum k -cut problem with given sizes of parts and the maximum coverage problem with a knapsack constraint — with preservation of the same performance guarantees in both cases.

To elucidate the key ideas behind the method we describe it informally under some general assumptions (of course, these are far from most general).

Assume that the problem under consideration can be formulated as the following nonlinear binary program:

$$\max F(x_1, \dots, x_n) \tag{1}$$

$$\text{s. t. } \sum_{i=1}^n x_i = p \tag{2}$$

$$x_i \in \{0, 1\}, \quad i = 1, \dots, n \tag{3}$$

where p is a positive integer, $F(x_1, \dots, x_n)$ is a function defined on the rational points of the n -dimensional cube $[0, 1]^n$ and computable in polynomial time. Assume further that one can associate with $F(x_1, \dots, x_n)$ another function $L(x_1, \dots, x_n)$ which is defined and polynomially computable on the same set, coincides with $F(x_1, \dots, x_n)$ on binary vectors, and the problem of maximizing $L(x_1, \dots, x_n)$ over all rational points of $[0, 1]^n$ subject to (2) is polynomially solvable. We shall call such a problem a *nice relaxation*. In our algorithms each nice relaxation will be in fact a slight reformulation of a linear program. Assume that additionally the following properties hold:

(A) there exists $C > 0$ such that $F(x_1, \dots, x_n) \geq CL(x_1, \dots, x_n)$ for each $x \in [0, 1]^n$;

(B) the function $\varphi(\varepsilon, x, i, j) = F(x_1, \dots, x_i + \varepsilon, \dots, x_j - \varepsilon, \dots, x_n)$ is convex with respect to $\varepsilon \in [-\min\{x_i, 1 - x_j\}, \min\{1 - x_i, x_j\}]$ for each pair of indices i and j and each $x \in [0, 1]^n$.

We claim that under the above assumptions one can find in polynomial time a feasible solution $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_n)$ to the problem (1)–(3), satisfying $F(\tilde{x}) \geq CF(x^*)$ where x^* is an optimal solution to (1)–(3). Let x' be an optimal solution to the nice relaxation. Indeed, if the vector x' is not binary, then due to (2) it has at least two different components x_i and x_j with values lying strictly between 0 and 1. By property (B), $\varphi(\varepsilon, x', i, j) \geq F(x')$ either for $\varepsilon = \min\{1 - x_i, x_j\}$, or for $\varepsilon = -\min\{x_i, 1 - x_j\}$. Thus we obtain a new feasible solution $x'' = (x'_1, \dots, x'_i + \varepsilon, \dots, x'_j - \varepsilon, \dots, x'_n)$ with smaller number of noninteger components and such that $F(x'') \geq F(x')$. After repeating this “pipage” step at

most n times we arrive at a binary feasible solution $\tilde{x}$ with $F(\tilde{x}) \geq CL(x') \geq CL(x^*) = CF(x^*)$, as required. Since an optimal solution to the nice relaxation can be found in polynomial time, the overall running time of the above described C-approximation algorithm is polynomially bounded.

2 The Maximum Coverage Problem

In the maximum coverage problem (MCP for short), given a family $\mathcal{F} = \{S_j : j \in J\}$ of subsets of a set $I = \{1, \dots, n\}$ with associated nonnegative weights w_j and a positive integer p , it is required to find a subset $X \subseteq I$ with $|X| = p$ so as to maximize the total weight of the sets in $\mathcal{F}$ having nonempty intersections with X . The polynomial-time solvability of MCP clearly implies that of the set cover problem and so it is NP -hard. In a sense MCP can be treated as an inverse of the set cover problem and like the latter has numerous applications (see, e. g. [10]). It is well known that a simple greedy algorithm solves MCP approximately within a factor of $1 - (1 - 1/p)^p$ of the optimum (Cornuejols, Fisher and Nemhauser [2]). Feige [4] proves that no polynomial algorithm can have better performance guarantee provided that $P \neq NP$. Another result concerning MCP is due to Cornuejols, Nemhauser and Wolsey [3] who prove that the greedy algorithm almost always finds an optimal solution to MCP in the case of two-element sets. We show below that MCP can be solved in polynomial time approximately within a factor of $1 - (1 - 1/k)^k$ of the optimum, where $k = \max\{|S_j| : j \in J\}$. Although $1 - (1 - 1/k)^k$ like $1 - (1 - 1/p)^p$ can be arbitrary close to $1 - e^{-1}$, the parameter k looks more interesting: for each fixed k ($k = 2, 3, \dots$) MCP still remains NP -hard. E. g., in the case when $k = 2$, which is the converse of the vertex cover problem, the performance guarantee of the greedy algorithm has the same value of $1 - e^{-1}$ [3], whereas our algorithm finds a solution within a factor of $3/4$ of the optimum. Ultimately, the performance guarantee of our algorithm beats the performance guarantee of the greedy algorithm in each case of bounded k and coincides with that when k is unbounded. Note also that our result is similar in a sense to the well-known result [9] that the set cover problem can be approximated in polynomial time within a factor of r of the optimum, where r is the maximum number of sets containing an element.

Let $J = \{1, \dots, m\}$. MCP can be equivalently reformulated as a constrained version of MAX SAT over variables $y_1, \dots, y_n$ with m clauses $C_1, \dots, C_m$ such that C_j is the collection of y_i with $i \in S_j$ and has weight w_j . It is required to assign “true” values to exactly p variables so as to maximize the total weight of satisfied clauses. Furthermore, analogously to MAX SAT (see, e. g. [6]), MCP can be stated as the following integer program:

$$\max \sum_{j=1}^m w_j z_j \quad (4)$$

$$\text{s. t.} \quad \sum_{i \in S_j} x_i \geq z_j, \quad j = 1, \dots, m, \quad (5)$$

$$\sum_{i=1}^n x_i = p, \quad (6)$$

$$x_i \in \{0, 1\}, \quad i = 1, \dots, n, \quad (7)$$

$$0 \leq z_i \leq 1, \quad i = 1, \dots, m. \quad (8)$$

It is easy to see that the relation “ $x_i = 1$ if $i \in X$, and $x_i = 0$ otherwise” establishes a 1-1 correspondence between the optimal sets in MCP and the optimal solutions to (4)–(8). Note that the variables x_i determine the optimal values of z_j in any optimal solution and represent active variables in the sense above. Moreover, it is clear that MCP is equivalent to maximizing the function $F(x) = \sum_{j=1}^m w_j (1 - \prod_{i \in S_j} (1 - x_i))$ over all binary vectors x satisfying (6). Observe also that the objective function (4) can be replaced by the function

$$L(x) = \sum_{j=1}^m w_j \min\{1, \sum_{i \in S_j} x_i\}$$

of the active variables $x_1, \dots, x_n$, thus providing a nice relaxation.

We now show that the functions F and L just defined satisfy properties (A) and (B).

Property (A) holds with $C = (1 - (1 - 1/k)^k)$ where $k = \max\{|S_j| : j \in J\}$, which is implied by the following inequality (used first by Goemans and Williamson [6] in a similar context):

$$1 - \prod_{i=1}^k (1 - y_i) \geq (1 - (1 - 1/k)^k) \min\{1, \sum_{i=1}^k y_i\}, \quad (9)$$

valid for all $0 \leq y_i \leq 1, i = 1, \dots, k$. To make the paper self-contained we derive it below. By using the arithmetic-geometric mean inequality we have that

$$1 - \prod_{i=1}^k (1 - y_i) \geq 1 - \left(1 - \frac{z}{k}\right)^k,$$

where $z = \min\{1, \sum_{i=1}^k y_i\}$. Since the function $g(z) = 1 - (1 - z/k)^k$ is concave on the segment $[0, 1]$ and $g(0) = 0, g(1) = 1 - (1 - 1/k)^k$, we finally obtain

$$g(z) \geq \left(1 - (1 - 1/k)^k\right) z,$$

as desired.

To check property (B) it suffices to observe that in this case the function $\varphi(\varepsilon, x, i, j)$ is convex because it is a quadratic polynomial in ε , whose main coefficient is nonnegative for each pair of indices i and j and each $x \in [0, 1]^n$. Thus by concretizing the general scheme described in the introduction we obtain a $(1 - (1 - 1/k)^k)$ -approximation algorithm for the maximum coverage problem.

We now demonstrate that the integrality gap of (4)–(8) can be arbitrarily close to $(1 - (1 - 1/k)^k)$ and thus the rounding scheme described above is best

possible for the integer program (4)–(8). Set $n = kp$, $w_j = 1$ for all j and let $\mathcal{F}$ be the collection of all subsets of $\{1, \dots, n\}$ with cardinality k . Then, by symmetry, any binary vector with exactly p units maximizes $L(x)$ subject to (6)–(7) and so the optimal value of this problem is equal to $L^* = C_n^k - C_{n-p}^k$. On the other hand, the vector with all components equal to $1/k$ provides an optimal solution of weight $L' = C_n^k$ to the linear relaxation in which the objective is to maximize $L(x)$ subject to (6) and $0 \leq x_i \leq 1$ for all i . Now it is easy to derive an upper bound on the ratio

$$\begin{aligned}
\frac{L^*}{L'} &= \frac{C_n^k - C_{n-p}^k}{C_n^k} \\
&= 1 - \frac{(n-p)!}{k!(n-p-k)!} \frac{k!(n-k)!}{n!} \\
&= 1 - \left(\frac{n-p}{n}\right) \left(\frac{n-p-1}{n-1}\right) \dots \left(\frac{n-p-k+1}{n-k+1}\right) \\
&\leq 1 - \left(\frac{n-p}{n}\right) \left(\frac{n-p-1}{n}\right) \dots \left(\frac{n-p-k+1}{n}\right) \\
&= 1 - \left(1 - \frac{1}{k}\right) \left(1 - \frac{1}{k} - \frac{1}{n}\right) \left(1 - \frac{1}{k} - \frac{2}{n}\right) \dots \left(1 - \frac{1}{k} - \frac{k+1}{n}\right) \\
&\leq 1 - \left(1 - \frac{1}{k} - \frac{k+1}{n}\right)^k,
\end{aligned}$$

which tends to $(1 - (1 - 1/k)^k)$ when k is fixed and $n \rightarrow \infty$.

Remark 1. The algorithm and the argument above can be easily adopted to yield the same performance guarantee in the case of the more general problem in which the constraint (6) is replaced by the constraints

$$\sum_{i \in I_t} x_i = p_t, \quad t = 1, \dots, r$$

where $\{I_t : t = 1, \dots, r\}$ is a partition of the ground set I and p_t are positive integers. It can be shown, on the other hand, that the worst-case ratio of the straightforward extension of the greedy algorithm cannot be lower bounded by any absolute constant. So, our algorithm is the only algorithm with constant performance guarantee among those known for this generalization.

Remark 2. It can be easily observed that from the very beginning (and with the same ultimate result) we could consider objective functions of the following more general form:

$$F(x) = \sum_{j=1}^m w_j \left(1 - \prod_{i \in S_j} (1 - x_i)\right) + \sum_{t=1}^l u_t \left(1 - \prod_{i \in R_t} x_i\right),$$

where S_j and R_t are arbitrary subsets of $\{1, \dots, n\}$, and u_t, w_j are nonnegative weights. The problem with such objective functions can be reformulated as the constrained MAX SAT in which each clause either contains no negations or contains nothing but negations.

3 The Maximum Cut Problem with Given Sizes of Parts

Let G be a complete undirected graph with vertex set $V(G) = \{1, \dots, n\}$. Any nonempty vertex subset $X \subseteq V(G)$ determines a cut $\delta(X)$ of G which, by definition, consists of the set of edges having exactly one end in X . Assume that to each edge $e = ij$ of G is assigned a nonnegative weight w_{ij} . In the maximum cut problem with given sizes of parts (MCGS for short), given a complete graph G , nonnegative edge weights w_{ij} and a positive integer $p \leq n/2$, it is required to find a cut $\delta(X)$ having maximum total weight over all cuts with $|X| = p$. In the special case of $p = n/2$, also known as the max bisection problem, an approximate solution of weight within a factor of $1/2$ of the optimum can be found in polynomial time by a simple random sampling algorithm. However, natural extensions of this algorithm to the case of arbitrary p do not admit any fixed constant performance guarantee. Frieze and Jerrum [5] prove that the max bisection problem can be approximately solved within a factor of 0.65 of the optimum by a randomized rounding of an optimal solution to a semidefinite program but their algorithm does not admit straightforward extensions. We present an approximation algorithm which finds a feasible solution to MCGS of weight within a factor of $1/2$ of the optimum in the case of arbitrary p .

Observe first that MCGS is equivalent to maximizing the function

$$F(x) = \sum_{i < j} w_{ij}(x_i + x_j - 2x_i x_j)$$

over all binary vectors x with exactly p units. Notice first that the function F has property (B) because of the same reason as in the above section. Consider the following integer program:

$$\max \sum_{i < j} w_{ij} z_{ij} \tag{10}$$

$$\text{s. t. } z_{ij} \leq x_i + x_j, \quad i < j, \tag{11}$$

$$z_{ij} \leq 2 - x_i - x_j, \quad i < j, \tag{12}$$

$$\sum_{i \in V(G)} x_i = p, \tag{13}$$

$$x_i, z_{ij} \in \{0, 1\}, \quad i \in V(G), \quad i < j. \tag{14}$$

It is easy to check that $X^* \subseteq V(G)$ provides an optimal cut in MCGS if and only if the vector x defined by $x_i^* = 1$ if $i \in X^*$ and 0 otherwise, is an optimal solution to (10)–(14). Note that again, variables x_i can be considered as active in the sense that the program (10)–(14) is in fact equivalent to

$$\max L(x) = \sum_{i < j} w_{ij} \min\{x_i + x_j, 2 - x_i - x_j\} \tag{15}$$

$$\text{s. t. } \sum_{i \in V(G)} x_i = p, \tag{16}$$

$$x_i \in \{0, 1\}, \quad i \in V(G). \tag{17}$$

Here we take as a nice relaxation the continuous relaxation of (15)–(17) (in which (17) is replaced by $0 \leq x_i \leq 1$ for all $i \in V(G)$). Let x be an optimal solution to the nice relaxation. We claim that $F(x) \geq 1/2L(x)$. Indeed, it suffices to check that

$$x_i + x_j - 2x_i x_j \geq 1/2 \min\{x_i + x_j, 2 - x_i - x_j\} \quad (18)$$

for all pairs i, j with $i < j$. Fix some $i < j$ and assume first that $x_i + x_j \leq 1$. Then (18) is equivalent to $x_i + x_j \geq 4x_i x_j$, which follows from

$$x_i + x_j \geq (x_i + x_j)^2 \geq 4x_i x_j.$$

The case when $x_i + x_j \geq 1$ is quite symmetric to the above, which is easily shown by the substitution $x_i = 1 - y_i$, $x_j = 1 - y_j$. Thus property (A) with $C = 1/2$ does hold and we obtain an $1/2$ -approximation algorithm for MCGS.

Consider an instance of MCGS in which $2p \leq n$, $w_{ij} = 1$ if both i and j lie in a fixed subset of $V(G)$ having cardinality $2p$, and $w_{ij} = 0$ otherwise. The optimal value of this instance is obviously equal to p^2 , whereas the optimal value of the linear relaxation is $p(p-1)/2$. Hence

$$\frac{p^2}{p(p-1)/2} = \frac{2p^2}{p^2 - p} = \frac{2}{1 - 1/p}$$

which can be arbitrarily close to 2. Thus, again, our rounding scheme is best possible for the integer program (10)–(14).

4 The Maximum k -Cut Problem with Given Sizes of Parts

In this section we illustrate how the “pipage” technique can be extended to apply to a natural generalization of MCGS.

In the maximum k -cut problem with given sizes of parts (k -MCGS), given a complete undirected graph G with $V(G) = \{1, \dots, n\}$, nonnegative edge weights w_{ij} , and a collection of nonnegative numbers $\{p_1, \dots, p_k\}$, it is required to find a partition $\{P_1, \dots, P_k\}$ of $V(G)$ with $|P_t| = p_t$ for each t , maximizing the function

$$\sum_{r < s} \sum_{i \in P_r} \sum_{j \in P_s} w_{ij}.$$

Two similar problems have been considered earlier by Guttmann-Beck and Hassin in [7] and [8]. The minimization version of k -MCGS, which is also NP-hard, is studied in [8]. No constant-factor algorithm for this problem is known. Guttmann-Beck and Hassin present a polynomial-time 3-approximation algorithm for the special case when the edge weights satisfy triangle inequality and the number of parts is fixed. The other paper [7] considers the min-sum p -clustering problem which can be treated as a complement to k -MCGS. The

authors suggest a 2-approximation algorithm for the similar special case of this problem.

We present further an approximation algorithm which produces a feasible solution to k -MCGS of weight within a factor of $1/2$ of the optimum. The problem can be reformulated as the following nonlinear integer program:

$$\max \quad F(x) = \sum_{i < j} w_{ij} \left(1 - \sum_{t=1}^k x_{it} x_{jt}\right) \quad (19)$$

$$\text{s. t.} \quad \sum_{i=1}^n x_{it} = p_t, \quad t = 1, \dots, k, \quad (20)$$

$$\sum_{t=1}^k x_{it} = 1, \quad i \in V(G), \quad (21)$$

$$x_{it} \in \{0, 1\}, \quad i \in V(G). \quad (22)$$

It is easy to check that x_{it}^* is an optimal solution to (19)–(22) if and only if $\{P_1, \dots, P_k\}$ with $P_t = \{i : x_{it} = 1\}$ for each t , is an optimal partition in k -MCGS.

Consider now the following linear integer program also equivalent to k -MCGS and virtually generalizing (10)–(14):

$$\max \quad \frac{1}{2} \sum_{i < j} w_{ij} \sum_{t=1}^k z_{ij}^t \quad (23)$$

$$\text{s. t.} \quad z_{ij}^t \leq x_{it} + x_{jt}, \quad i < j, \quad t = 1, \dots, k, \quad (24)$$

$$z_{ij}^t \leq 2 - x_{it} - x_{jt}, \quad i < j, \quad t = 1, \dots, k, \quad (25)$$

$$\sum_{i=1}^n x_{it} = p_t, \quad t = 1, \dots, k, \quad (26)$$

$$\sum_{t=1}^k x_{it} = 1, \quad i \in V(G), \quad (27)$$

$$x_{it}, z_{ij}^t \in \{0, 1\}, \quad i, j \in V(G), \quad t = 1, \dots, k. \quad (28)$$

As in the previous section the objective function (23) can be replaced by

$$L(x) = \frac{1}{2} \sum_{i < j} w_{ij} \sum_{t=1}^k \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\}.$$

We claim that $F(x) \geq 1/2 L(x)$ for all feasible $x = (x_{it})$. Fix a pair of indices i and j with $i < j$. It suffices to show that

$$1 - \sum_{t=1}^k x_{it} x_{jt} \geq 1/4 \sum_{t=1}^k \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\}. \quad (29)$$

Indeed, we have already proved in the above section (see (18)) that for each t

$$x_{it} + x_{jt} - 2x_{it}x_{jt} \geq 1/2 \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\}. \quad (30)$$

Adding together (30) over t from 1 to k and taking into account (27) we obtain that

$$2 - 2 \sum_{t=1}^k x_{it}x_{jt} \geq 1/2 \sum_{t=1}^k \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\},$$

which implies (29).

We describe now the “pipage” step. Let x be an optimal solution to the linear relaxation of (23)–(28) (that is, with (28) replaced by $0 \leq x_{jt} \leq 1$, $0 \leq z_{ij}^t \leq 1$). Define the bipartite graph H with the bipartition $(\{1, \dots, n\}, \{1, \dots, k\})$ so that $jt \in E(H)$ if and only if x_{jt} is fractional. Note that (26) and (27) imply that each vertex of H is either isolated or has degree at least 2. Assume that x has fractional components. Since H is bipartite it follows that H has a circuit C of even length. Let M_1 and M_2 be the matchings of H whose union is the circuit C . Define a new solution $x(\varepsilon)$ by the following rule: if jt is not an edge of C , then $x_{jt}(\varepsilon)$ coincides with x_{jt} , otherwise, $x_{jt}(\varepsilon) = x_{jt} + \varepsilon$ if $jt \in M_1$, and $x_{jt}(\varepsilon) = x_{jt} - \varepsilon$ if $jt \in M_2$.

By definition $x(\varepsilon)$ is a feasible solution to the linear relaxation of (23)–(28) for all $\varepsilon \in [-\varepsilon_1, \varepsilon_2]$ where

$$\varepsilon_1 = \min\left\{\min_{jt \in M_1} x_{jt}, \min_{jt \in M_2} (1 - x_{jt})\right\}$$

and

$$\varepsilon_2 = \min\left\{\min_{jt \in M_1} (1 - x_{jt}), \min_{jt \in M_2} x_{jt}\right\}.$$

Moreover, $F(x(\varepsilon))$ is a quadratic polynomial in ε with a nonnegative main coefficient and therefore

$$F(x(\varepsilon^*)) \geq F(x) \geq 1/2L(x)$$

for some $\varepsilon^* \in \{-\varepsilon_1, \varepsilon_2\}$. The new solution $x(\varepsilon^*)$, being feasible for the linear relaxation of (23)–(28), has a smaller number of fractional components. Set $x' = x(\varepsilon^*)$ and, if x' has fractional components, apply to x' the above described “pipage” step and so on. Ultimately, after at most nk steps, we arrive at a solution $\tilde{x}$ which is feasible for (19)–(22) and satisfies

$$F(\tilde{x}) \geq 1/2L(x) \geq 1/2F^*,$$

where F^* is an optimal value of (19)–(22) (and of the original instance of k -MCGS). Thus we have designed an $1/2$ -approximation algorithm for k -MCGS.

5 The Maximum Coverage Problem with a Knapsack Constraint

In this section we show that the “pipage” rounding of linear relaxations in conjunction with the partial enumeration method developed by Sahni [12] for the

knapsack problem can yield good constant-factor approximation algorithms even in the case of a knapsack constraint.

The maximum coverage problem with a knapsack constraint (MCKP) is, given a family $\mathcal{F} = \{S_j : j \in J\}$ of subsets of a set $I = \{1, \dots, n\}$ with associated nonnegative weights w_j and costs c_j , and a positive integer B , to find a subset $X \subseteq I$ with $\sum_{j \in X} c_j \leq B$ so as to maximize the total weight of the sets in $\mathcal{F}$ having nonempty intersections with X . MCKP includes both MCP and the knapsack problem as special cases. Wolsey [13] appears to be the first who succeeded in constructing a constant-factor approximation algorithm for MCKP even in a more general setting with an arbitrary nondecreasing submodular objective function. His algorithm is of greedy type and has performance guarantee of $1 - e^{-\beta} \approx 0.35$ where β is the root of the equation $e^\beta = 2 - \beta$. Recently, Khuller, Moss and Naor [11] have designed a $(1 - e^{-1})$ -approximation algorithm for MCKP by combining the partial enumeration method of Sahní for the knapsack problem [12] with a simple greedy procedure. In this section we present an $1 - (1 - 1/k)^k$ -approximation algorithm for MCKP where k is the maximum size of sets in the instance. Our algorithm exploits the same idea of partial enumeration but instead finding a greedy solution, solves a linear relaxation and then rounds the fractional solution by a bit more general “pipage” procedure.

Generalizing (4)–(8) rewrite MCKP as the following integer program:

$$\max \sum_{j=1}^m w_j z_j \quad (31)$$

$$\text{s. t. } \sum_{i \in S_j} x_i \geq z_j, \quad j \in J, \quad (32)$$

$$\sum_{i=1}^n c_i x_i \leq B, \quad (33)$$

$$0 \leq x_i, z_j \leq 1, \quad i \in I, j \in J \quad (34)$$

$$x_i \in \{0, 1\}, \quad i \in I. \quad (35)$$

Note that one can exclude variables z_j by rewriting (31)–(35) as the following equivalent nonlinear program:

$$\max \quad F(x) = \sum_{j=1}^m w_j \left(1 - \prod_{i \in S_j} (1 - x_i)\right) \quad (36)$$

subject to (33)–(35).

Set $k = \max\{|S_j| : j \in J\}$. Denote by $IP[I_0, I_1]$ and $LP[I_0, I_1]$ the integer program (31)–(35) and its linear relaxation (31)–(34) respectively, subject to the additional constraints: $x_i = 0$ for $i \in I_0$ and $x_i = 1$ for $i \in I_1$ where I_0 and I_1 are disjoint subsets of I . By a solution to $IP[I_0, I_1]$ and $LP[I_0, I_1]$ we shall mean only a vector x , since the optimal values of z_j are trivially computed if x is fixed.

We first describe an auxiliary algorithm $\mathcal{A}$ which finds a feasible solution $x^{\mathcal{A}}$ to the linear program $LP[I_0, I_1]$. The algorithm is divided into two phases.

The first phase consists in finding an optimal solution x^{LP} to $LP(I_0, I_1)$ by application one of the known polynomial-time algorithms. The second phase of $\mathcal{A}$ transforms x^{LP} into $x^{\mathcal{A}}$ through a series of “pipage” steps. We now describe the general step. Set $x^{\mathcal{A}} \leftarrow x^{LP}$. If at most one component of $x^{\mathcal{A}}$ is fractional, stop. Otherwise, choose two indices i' and i'' such that $0 < x_{i'}^{\mathcal{A}} < 1$ and $0 < x_{i''}^{\mathcal{A}} < 1$. Set $x_{i'}^{\mathcal{A}}(\varepsilon) \leftarrow x_{i'}^{\mathcal{A}} + \varepsilon$, $x_{i''}^{\mathcal{A}}(\varepsilon) \leftarrow x_{i''}^{\mathcal{A}} - \varepsilon c_{i'}/c_{i''}$ and $x_k^{\mathcal{A}}(\varepsilon) \leftarrow x_k^{\mathcal{A}}$ for all $k \neq i', i''$. Find an endpoint ε^* of the interval

$$[-\min\{x_{i'}, (1 - x_{i''}) \frac{c_{i''}}{c_{i'}}\}, \min\{1 - x_{i'}, x_{i''} \frac{c_{i''}}{c_{i'}}\}].$$

such that $F(x(\varepsilon^*)) \geq F(x^{\mathcal{A}})$. Set $x^{\mathcal{A}} \leftarrow x(\varepsilon^*)$. Go to the next step.

The correctness of the second phase follows from the fact that the vector $x(\varepsilon)$ is feasible for each ε in the above interval, and from the earlier observation (see Section 2) that the function $F(x(\varepsilon))$ is convex.

Each “pipage” step of the second phase of $\mathcal{A}$ reduces the number of fractional components of the current vector $x^{\mathcal{A}}$ at least by one. So, finally, $\mathcal{A}$ outputs an “almost integral” feasible vector $x^{\mathcal{A}}$ having at most one fractional component.

By construction, $F(x^{\mathcal{A}}) \geq F(x^{LP})$. By (9), it follows that

$$\begin{aligned} F(x^{\mathcal{A}}) &\geq F(x^{LP}) \\ &\geq \sum_{j \in J_1} w_j + (1 - (1 - 1/k)^k) \sum_{j \in J \setminus J_1} w_j \min\{1, \sum_{i \in S_j} x_i^{LP}\} \end{aligned} \quad (37)$$

where and henceforth $J_1 = \{j : S_j \cap I_1 \neq \emptyset\}$.

We now present a description of the whole algorithm.

Input: An instance of the integer program (31)–(35);

Output: A feasible solution $\bar{x}$ to the instance;

Among all feasible solutions x to the instance satisfying $\sum_{i \in I} x_i \leq 3$, by complete enumeration find a solution x^0 of maximum weight;

$\bar{x} \leftarrow x^0$;

for all $I_1 \subset I$ such that $|I_1| = 4$ and $\sum_{i \in I_1} c_i \leq B$ **do**

begin

$I_0 \leftarrow \emptyset$;

$t \leftarrow 0$;

while $t = 0$ **do**

begin

 apply $\mathcal{A}$ to $LP[I_0, I_1]$;

if all $x_i^{\mathcal{A}}$ are integral

then begin $t \leftarrow 1$; $\hat{x} \leftarrow x^{\mathcal{A}}$ **end**

otherwise

begin

 find i' such that $x_{i'}^{\mathcal{A}}$ is fractional;

$\hat{x}_{i'} \leftarrow 0$;

for all $i \neq i'$ **do** $\hat{x}_i \leftarrow x_i^{\mathcal{A}}$;

$I_0 \leftarrow I_0 \cup \{i'\}$

end

end

if $F(\hat{x}) > F(\bar{x})$ **then** $\bar{x} \leftarrow \hat{x}$
end
end

We now prove that the performance guarantee of the described algorithm is indeed $(1 - (1 - 1/k)^k)$.

Let X^* be an optimal set of the given instance of MCKP. Denote by x^* the incidence vector of X^* . Recall that x^* is an optimal solution to the equivalent nonlinear program (36), (33)–(35). If $|X^*| \leq 3$, then the output of the algorithm is an optimal solution. So we may assume that $|X^*| \geq 4$. W.l.o.g. we may also assume that the set I is ordered in such a way that $X^* = \{1, \dots, |X^*|\}$ and for each $i \in X^*$, the element i covers the sets in $\mathcal{F}$ not covered by the elements $1, \dots, i-1$ of the maximum total weight among $j \in X^* \setminus \{1, \dots, i-1\}$.

Consider now that iteration of the algorithm at which $I_1 = \{1, 2, 3, 4\}$. At this iteration the algorithm runs through q steps, $q \leq n-4$. At step t it calls the algorithm $\mathcal{A}$ to find an “almost integral” feasible solution $x^t = x^{\mathcal{A}}$ to $IP[I_0^t, I_1]$ where $I_0^t = \{i_1, \dots, i_{t-1}\}$. If all components of x^t are integral then $t = q$ and the iteration ends up. Otherwise the vector x^t is replaced by the integral vector $\hat{x}^t$ which coincides with x^t in its integral components and equals zero in its single fractional component indexed by i_t . If the weight of $\hat{x}^t$ exceeds that of the record solution, the latter is updated. Then the algorithm sets $I_0^{t+1} = I_0^t \cup \{i_t\}$ and goes to the next step. Thus at the iteration the algorithm finds a series of feasible integral solutions $\hat{x}^1, \dots, \hat{x}^q$ to (36), (33)–(35) or, equivalently, subsets $\hat{X}_1, \dots, \hat{X}_q \subseteq I$ satisfying $\hat{X}_t \supseteq I_1$ and $\hat{X}_t \cap I_0^t = \emptyset$ where $I_0^t = \{i_1, \dots, i_{t-1}\}$, $t = 1, \dots, q$.

Assume first that $X^* \cap I_0^q = \emptyset$. Then x^* is an optimal solution to $IP[I_0^q, I_1]$. Recall that $\hat{x}^q = x^q$ and the latter is obtained from the optimal solution to $LP[I_0^q, I_1]$ by the “pipage” process. By (37) it follows that the solution $\hat{x}^q$, being not better than the output solution, has weight within a factor of $(1 - (1 - 1/k)^k)$ of the optimum. Thus, in this case we are done.

Assume now that $X^* \cap I_0^q \neq \emptyset$ and let I_0^{s+1} be the first set in the series $I_0^1 = \emptyset, \dots, I_0^q$, having nonempty intersection with X^* . In other words, i_s is the first index in the series $i_1, \dots, i_{q-1}$ lying in X^* . We claim then that

$$F(\hat{x}^s) \geq (1 - (1 - 1/k)^k) F(x^*). \quad (38)$$

Since the weight of $\hat{x}^s$ does not exceed the weight of the output vector of the algorithm this would prove that $(1 - (1 - 1/k)^k)$ is the performance guarantee of the algorithm.

In the following argument for brevity we shall use alternately the sets and their incidence vectors as the arguments of F .

Indeed, the function F can be also treated as the set function $F(X)$ defined on all subsets $X \subseteq I$. It is well known that $F(X)$ is submodular and, consequently, have the property that

$$F(X \cup \{i\}) - F(X) \geq F(X \cup Y \cup \{i\}) - F(X \cup Y), \quad (39)$$

for all $X, Y \subseteq I$ and $i \in I$. Let $i \in I$ and $Y \supseteq I_1$. Then

$$\begin{aligned}
1/4F(I_1) &= 1/4F(\{1, 2, 3, 4\}) = 1/4(F(\{1, 2, 3, 4\}) - F(\{1, 2, 3\}) + \\
&\quad F(\{1, 2, 3\}) - F(\{1, 2\}) + \\
&\quad F(\{1, 2\}) - F(\{1\}) + \\
&\quad F(\{1\}) - F(\emptyset)) \\
&\quad (\text{by the choice of } I_1) \\
&\geq 1/4(F(\{1, 2, 3, i\}) - F(\{1, 2, 3\}) + \\
&\quad F(\{1, 2, i\}) - F(\{1, 2\}) + \\
&\quad F(\{1, i\}) - F(\{1\}) + \\
&\quad F(\{i\}) - F(\emptyset)) \\
&\quad (\text{by (39)}) \\
&\geq (F(Y \cup \{i\}) - F(Y)).
\end{aligned}$$

Thus, we have proved that for any $Y \supseteq I_1$ and any $i \in I$,

$$1/4F(I_1) \geq F(Y \cup \{i\}) - F(Y). \quad (40)$$

Recall that the integral vector $\hat{x}^s$ is obtained from an “almost integral” vector x^s returned by the algorithm $\mathcal{A}$, by the replacement with zero its single fractional component $x_{i_s}^s$. It follows that

$$F(\hat{X}_s \cup \{i_s\}) \geq F(x_s). \quad (41)$$

Let x^{LP}, z^{LP} denote an optimal solution to $LP[I_0^s, I_1]$. Using (35), (40) and (41) we finally obtain

$$\begin{aligned}
F(\hat{x}^s) &= F(\hat{X}_s) \\
&= F(\hat{X}_s \cup \{i_s\}) - (F(\hat{X}_s \cup \{i_s\}) - F(\hat{X}_s)) \\
&\quad (\text{by (40)}) \\
&\geq F(\hat{X}_s \cup \{i_s\}) - 1/4F(I_1) \\
&\quad (\text{by (41)}) \\
&\geq F(x^s) - 1/4F(I_1) \\
&= \sum_{j \in J_1} w_j + \sum_{j \in J \setminus J_1} w_j (1 - \prod_{i \in S_j} (1 - x_i^s)) - 1/4 \sum_{j \in J_1} w_j \\
&\quad (\text{by (37)}) \\
&\geq 3/4 \sum_{j \in J_1} w_j + (1 - (1 - 1/k)^k) \sum_{j \in J \setminus J_1} w_j \min\{1, \sum_{i \in S_j} x_i^{LP}\} \\
&\geq (1 - (1 - 1/k)^k) \left(\sum_{j \in J_1} w_j + \sum_{j \in J \setminus J_1} w_j \min\{1, \sum_{i \in S_j} x_i^{LP}\} \right)
\end{aligned}$$

(by the choice of s)

$$\begin{aligned} &\geq (1 - (1 - 1/k)^k) \left(\sum_{j \in J_1} w_j + \sum_{j \in J \setminus J_1} w_j \min\{1, \sum_{i \in S_j} x_i^*\} \right) \\ &= (1 - (1 - 1/k)^k) F(x^*). \end{aligned}$$

Acknowledgement

The authors wish to thank Refael Hassin for helpful comments and for pointing out some references.

References

1. N. Alon and J. H. Spenser, The probabilistic method, John Wiley and Sons, New York, 1992.
2. G. Cornuejols, M. L. Fisher and G. L. Nemhauser, Location of bank accounts to optimize float: an analytic study exact and approximate algorithms, *Management Science* 23 (1977) 789–810.
3. G. Cornuejols, G. L. Nemhauser and L. A. Wolsey, Worst-case and probabilistic analysis of algorithms for a location problem, *Operations Research* 28 (1980) 847–858.
4. U. Feige, A threshold of $\ln n$ for approximating set cover, *J. of ACM.* 45 (1998) 634–652.
5. A. Frieze and M. Jerrum, Improved approximation algorithms for MAX k -CUT and MAX BISECTION, *Algorithmica* 18 (1997) 67–81.
6. M. X. Goemans and D. P. Williamson, New $3/4$ -approximation algorithms for MAX SAT, *SIAM J. Discrete Math.* 7 (1994) 656–666.
7. N. Guttman-Beck and R. Hassin, Approximation algorithms for min-sum p -clustering, *Discrete Appl. Math.* 89 (1998) 125–142.
8. N. Guttman-Beck and R. Hassin, Approximation algorithms for minimum K -cut, to appear in *Algorithmica*.
9. D. S. Hochbaum, Approximation algorithms for the set covering and vertex cover problems, *SIAM J. on Computing* 11 (1982) 555–556.
10. D. S. Hochbaum, Approximating covering and packing problems: Set Cover, Vertex Cover, Independent Set, and related problems, in: D. S. Hochbaum, ed., *Approximation algorithms for NP-hard problems* (PWS Publishing Company, New York, 1997) 94–143.
11. S. Khuller, A. Moss and J. Naor, The budgeted maximum coverage problem, to appear in *Inform. Proc. Letters*.
12. S. Sahni, Approximate algorithms for the 0–1 knapsack problem, *J. of ACM* 22 (1975) 115–124.
13. L. A. Wolsey, Maximizing real-valued submodular functions: primal and dual heuristics for location problems, *Math. Oper. Res.* 7 (1982) 410–425.

Solving the Convex Cost Integer Dual Network Flow Problem

Ravindra K. Ahuja¹, Dorit S. Hochbaum², and James B. Orlin³

¹Industrial & Systems Engineering,
University of Florida, Gainesville, FL 32611, USA
Ahuja@ufl.edu

²Department of IE & OR and Haas School of Management,
University of California, Berkeley, CA 94720, USA
Dorit@hochbaum.IEOR.berkeley.edu

³Sloan School of Management, MIT,
Cambridge, MA 02139, USA
Jorlin@mit.edu

Abstract. In this paper, we consider a convex optimization problem where the objective function is the sum of separable convex functions, the constraints are similar to those arising in the dual of a minimum cost flow problem (that is, of the form $\pi(i) - \pi(j) \leq w_{ij}$), and the variables are required to take integer values within a specified range bounded by an integer U . Let m denote the number of constraints and $(n+m)$ denote the number of variables. We call this problem *the convex cost integer dual network flow problem*. In this paper, we develop network flow based algorithms to solve the convex cost integer dual network flow problem efficiently. We show that using the Lagrangian relaxation technique, the convex cost integer dual network flow problem can be reduced to a convex cost primal network flow problem where each cost function is a piecewise linear convex function with integer slopes. We next show that the cost scaling algorithm for the minimum cost flow problem can be adapted to solve the convex cost integer dual network flow problem in $O(nm \log(n^2/m) \log(nU))$ time. This algorithm improves the best currently available algorithm and is also likely to yield algorithms with attractive empirical performance.

1 Introduction

In this paper, we consider the following integer programming problem with convex costs:

$$\text{Minimize } \sum_{(i,j) \in Q} \bar{F}_{ij}(w_{ij}) + \sum_{i \in P} \bar{B}_i(\mu_i) \quad (1a)$$

subject to

$$\mu_i - \mu_j = w_{ij} \quad \text{for all } (i,j) \in Q, \quad (1b)$$

$$l_{ij} \leq w_{ij} \leq u_{ij} \quad \text{for all } (i, j) \in Q, \quad (1c)$$

$$l_i \leq \mu_i \leq u_i \quad \text{for all } i \in P, \quad (1d)$$

$$w_{ij} \text{ integer for all } (i, j) \in Q, \text{ and } \mu_i \text{ integer for all } i \in P, \quad (1e)$$

where $P = \{1, 2, \dots, n\}$ is a set of n numbers, $Q \subseteq P \times P$, and l_i , u_i , l_{ij} , and u_{ij} are specified integers. Let $m = |Q|$. In this problem, μ_i 's and w_{ij} 's are decision variables, $\bar{F}_{ij}(w_{ij})$ is a convex function for every $(i, j) \in Q$, and $\bar{B}_i(\mu_i)$ is also a convex function of μ_i for every $i \in P$. Let $U = \max[\max\{u_{ij} - l_{ij} : (i, j) \in Q\}, \max\{u_i - l_i : i \in P\}]$.

We call problem (1) the *convex cost integer dual network flow problem* (or simply the *dual network flow problem*), because the constraints in (1b) are the constraints in the dual of the minimum cost flow problem. (As a matter of fact, the constraints arising in the dual of the minimum cost flow problem are of the form $\mu_i - \mu_j \leq w_{ij}$. We will show in Section 5 that the constraints of the form $\mu_i - \mu_j \leq w_{ij}$ can be transformed to the constraints $\mu_i - \mu_j = w_{ij}$.) The dual network flow problem and its special cases arise in various application settings, including multi-product multistage production/inventory systems, project scheduling, machine-scheduling with precedence constraints, dial-a-ride transit problems, inverse optimization, graph drawing of rectangles, and isotone regression. We describe several applications of the dual network flow problem in Section 6.

Since each function $\bar{F}_{ij}(w_{ij})$ is a convex function of the decision variable w_{ij} which is required to take integer values, we can replace $\bar{F}_{ij}(w_{ij})$ by a piecewise linear convex function $F_{ij}(w_{ij})$ with integer breakpoints (that is, we set $F_{ij}(w_{ij}) = \bar{F}_{ij}(w_{ij})$ for each integer w_{ij} in the range $[l_{ij}, u_{ij}]$ and make $F_{ij}(w_{ij})$ linear between every two consecutive integer values of w_{ij}). We perform the same transformation for each function $\bar{B}_i(\mu_i)$ to obtain a piecewise linear convex function $B_i(\mu_i)$. Now consider the following problem where we drop the integrality constraints on decision variables:

$$\text{Minimize } \sum_{(i,j) \in Q} F_{ij}(w_{ij}) + \sum_{i \in P} B_i(\mu_i) \quad (2a)$$

$$\mu_i - \mu_j = w_{ij} \quad \text{for all } (i, j) \in Q, \quad (2b)$$

$$l_{ij} \leq w_{ij} \leq u_{ij} \quad \text{for all } (i, j) \in Q, \quad (2c)$$

$$l_i \leq \mu_i \leq u_i \quad \text{for all } i \in P. \quad (2d)$$

It is well known that these always exist an optimal solution of (2) that is integer. It is also well known that (2) can further be transformed to a linear programming problem by introducing a separate variable for each linear segment in the functions $F_{ij}(w_{ij})$ and $B_i(\mu_i)$. This gives us a linear programming problem with $O(uU + mU)$ decision variables and hence can be solved in pseudo-polynomial time.

In this paper we develop an $O(nm \log(n^2/m) \log(nU))$ algorithm to solve the dual network flow problem. Rockafellar [1984] used a Lagrangian relaxation approach for solving the dual network flow problem. He showed that the Lagrangian multiplier problem is a minimum convex cost network flow problem. We use his transformation as our starting point for our approach. We solve the Lagrangian multiplier problem by

using an adaptation of the preflow-push cost-scaling algorithm for the minimum cost flow problem due to Goldberg and Tarjan [1987] and show that a solution to the Lagrangian multiplier can be used to solve the dual network flow problem.

Our paper makes the following contributions: (i) We show that the minimum convex cost network flow problem, obtained through Lagrangian relaxation technique, is a piecewise linear convex function with integer slopes. Integrality of the slopes allows us to solve this problem using Goldberg and Tarjan's cost scaling algorithm for the minimum cost flow problem. (The minimum convex cost network flow problem with general cost structure cannot be solved by Goldberg and Tarjan's cost scaling algorithm.) (ii) We use another special property of the cost function to implement the cost scaling algorithm in a manner so that the running time of the cost scaling algorithm is the same as for solving the linear cost minimum cost flow problem. The running time of our algorithm is $O(nm \log(n^2/m) \log(nU))$, which gives the best running time to solve the dual network flow problem. According to McCormick [1998], the cancel and tighten algorithm due to Karzanov and McCormick [1997] can be used to solve the convex cost integer dual network flow problem in $O(nm \log n \log(nU))$ time. Our algorithm improves this cancel and tighten approach for all network densities.

2 Transformation to a Network Flow Problem

We use the Lagrangian relaxation technique of Rockafellar [1984] to solve the dual network flow problem. He showed that the Lagrangian multiplier problem is a minimum convex cost network flow problem. We review his approach here and include some additional results. Our approach differs from his approach in a couple of aspects. First of all, we are focused on finding an optimal integral solution. Second, we are focused on developing formally efficient algorithms. We also include this material in this paper because our notation and basic network terminology are substantially different from that of Rockafellar.

As per Rockafellar [1984], we dualize the constraints (2b) using the vector x , obtaining the following Lagrangian subproblem:

$$\text{Minimize } \sum_{(i,j) \in Q} F_{ij}(w_{ij}) + \sum_{i \in P} B_i(\mu_i) - \sum_{(i,j) \in Q} (w_{ij} + \mu_j - \mu_i)x_{ij} \quad (3a)$$

subject to

$$l_{ij} \leq w_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in Q, \quad (3b)$$

$$l_i \leq \mu_i \leq u_i \quad \text{for all } i \in P. \quad (3c)$$

Notice that

$$\sum_{(i,j) \in Q} (\mu_j - \mu_i) x_{ij} = \sum_{i \in P} \mu_i \left(\sum_{\{j: (j,i) \in Q\}} x_{ji} - \sum_{\{j: (i,j) \in Q\}} x_{ij} \right) = \sum_{i \in P} \mu_i x_{i0}, \quad (4)$$

where $x_{i0} = \sum_{\{j: (j,i) \in Q\}} x_{ji} - \sum_{\{j: (i,j) \in Q\}} x_{ij}$ for each $i \in P$. Substituting (4) into (3) yields

$$L(x) = \min \sum_{(i,j) \in Q} \{ F_{ij}(w_{ij}) - x_{ij} w_{ij} \} + \sum_{i \in P} \{ B_i(\mu_i) - x_{i0} \mu_i \} \quad (5a)$$

subject to

$$x_{i0} = \sum_{\{j:(j,i) \in Q\}} x_{ji} - \sum_{\{j:(i,j) \in Q\}} x_{ij} \quad \text{for all } i \in P, \quad (5b)$$

$$l_{ij} \leq w_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in Q, \quad (5c)$$

$$l_i \leq \mu_i \leq u_i \quad \text{for all } i \in P. \quad (5d)$$

Since each $F_{ij}(w_{ij})$ is a piecewise linear convex function with integer breakpoints, $F_{ij}(w_{ij}) - x_{ij}w_{ij}$ is also a piecewise linear convex function with integer breakpoints. Similarly, each $B_i(\mu_i) - x_{i0}\mu_i$ is also a piecewise linear function with integer breakpoints. For a given value of x , each term in the first and second summation is a convex separable function and can be optimized separately. Consequently, in order to determine $L(x)$, we need to determine the following:

- (i) for each $(i,j) \in Q$ the value of w_{ij} , $l_{ij} \leq w_{ij} \leq u_{ij}$ for which $F_{ij}(w_{ij}) - x_{ij}w_{ij}$ attains its lowest value;
- (i) for each $i \in P$ the value of μ_i for which $B_i(\mu_i) - x_{i0}\mu_i$ attains its lowest value.

Any piecewise linear convex function with integer breakpoints defined over an interval $[\alpha, \beta]$ with α and β integers always has an integer optimal solution. Therefore, we can find the minimum value of $F_{ij}(w_{ij}) - x_{ij}w_{ij}$ by performing binary search over the integers in the interval $[l_{ij}, u_{ij}]$, which requires $O(\log U)$ time. Similarly, we can determine the minimum value of $B_i(\mu_i) - x_{i0}\mu_i$ in $O(\log U)$ time. Hence, we need a total of $O((n+m) \log U)$ time to determine $L(x)$.

We have thus shown that for a specified value of x , the Lagrangian subproblem can be efficiently solved. We now focus on solving the *Lagrangian Multiplier Problem*, which is to determine the value of x for which the Lagrangian subproblem attains the highest objective function value. The Lagrangian multiplier problem is to determine x^* such that

$$L(x^*) = \max L(x) \quad (6a)$$

subject to

$$x_{i0} = \sum_{\{j:(j,i) \in Q\}} x_{ji} - \sum_{\{j:(i,j) \in Q\}} x_{ij} \quad \text{for each } i \in P, \quad (6b)$$

where $L(x)$ is defined as in (5a). The following well-known theorem for convex optimization establishes a connection between the Lagrangian multiplier and the dual network flow problem.

Theorem 1. *Let x^* be an optimal solution of the Lagrangian multiplier problem (6). Then $L(x^*)$ equals the optimal objective function value of the dual network flow problem (2).*

Proof: The dual network flow problem (2) can be transformed to a linear programming problem by introducing a separate variable for each linear segment in the cost functions $F_{ij}(w_{ij})$ and $B_i(\mu_i)$. Then, Theorem 1 follows from a well known

result in the theory of Lagrangian relaxation (see, for example, Ahuja, Magnanti and Orlin [1993], Chapter 16). ♦

We will now focus on solving the Lagrangian multiplier problem (6). We will show that the Lagrangian multiplier problem can be transformed into a network flow problem with nonlinear arc costs. We define a directed network $G = (N, A)$ with the node set N and the arc set A . The node set N contains a node i for each element $i \in P$ and an extra node, node 0. The arc set A contains an arc (i, j) for each $(i, j) \in Q$ and an arc $(i, 0)$ for each $i \in P$. For each $(i, 0)$, $i \in P$, we let $w_{i0} = \mu_i$, $l_{i0} = l_i$, $\mu_{i0} = \mu_i$, and $F_{i0}(w_{i0}) = B_i(\mu_i)$. Let us define the function $H_{ij}(x_{ij})$ for each arc (i, j) in the following manner:

$$H_{ij}(x_{ij}) = \min\{F_{ij}(w_{ij}) - x_{ij}w_{ij} : l_{ij} \leq w_{ij} \leq u_{ij}\}. \quad (7)$$

In terms of this notation, the Lagrangian multiplier problem (6) can be restated as

$$\text{Maximize } \sum_{(i,j) \in A} H_{ij}(x_{ij}) \quad (8a)$$

subject to

$$\sum_{\{j:(j,i) \in A\}} x_{ji} - \sum_{\{j:(i,j) \in A\}} x_{ij} = 0 \quad \text{for all } i \in N, \quad (8b)$$

which is a network flow problem with nonlinear cost functions and with no upper or lower bounds on arc flows. In the next section, we will study properties of the function $H_{ij}(x_{ij})$.

3 Properties of the Function $H_{ij}(x_{ij})$

In this section, we show that for any arc (i, j) , $H_{ij}(x_{ij})$ is a piecewise linear concave function with at most $(u_{ij} - l_{ij} + 1)$ linear segments and the slopes of these segments are integers between $-u_{ij}$ and $-l_{ij}$. We shall use these properties in the next section to develop an efficient cost scaling algorithm to solve the Lagrangian multiplier problem. Let $b_{ij}(\theta) = F_{ij}(\theta + 1) - F_{ij}(\theta)$ for $l_{ij} \leq \theta \leq u_{ij} - 1$, θ integral. The fact that $F_{ij}(\theta)$ is a convex function of θ can be used to prove the following results:

Property 1. $b_{ij}(\theta) \leq b_{ij}(\theta + 1) \leq b_{ij}(\theta + 2) \leq \dots$

The following theorem obtains the expression for $H_{ij}(x_{ij})$:

Theorem 2. *The function $H_{ij}(x_{ij}) = \min\{F_{ij}(w_{ij}) - w_{ij}x_{ij} : l_{ij} \leq w_{ij} \leq u_{ij} \text{ and } w_{ij} \text{ integer}\}$ is a piecewise linear concave function of x_{ij} , and is described in the following manner:*

$$H_{ij}(x_{ij}) = \begin{cases} F_{ij}(l_{ij}) - l_{ij}x_{ij} & \text{if } -\infty \leq x_{ij} \leq b_{ij}(l_{ij}) \\ F_{ij}(l_{ij} + 1) - (l_{ij} + 1)x_{ij} & \text{if } b_{ij}(l_{ij}) \leq x_{ij} \leq b_{ij}(l_{ij} + 1) \\ \vdots & \\ F_{ij}(\theta) - \theta x_{ij} & \text{if } b_{ij}(\theta - 1) \leq x_{ij} \leq b_{ij}(\theta) \\ \vdots & \\ F_{ij}(u_{ij}) - u_{ij}x_{ij} & \text{if } b_{ij}(u_{ij} - 1) \leq x_{ij} \leq \infty \end{cases} \quad (9)$$

Proof Sketch: The function $H_{ij}(x_{ij})$ is defined as $H_{ij}(x_{ij}) = \min\{F_{ij}(w_{ij}) - x_{ij}w_{ij} : l_{ij} \leq w_{ij} \leq u_{ij} \text{ and integer}\}$, or $H_{ij}(x_{ij}) = \min\{F_{ij}(\theta) - \theta x_{ij} : l_{ij} \leq \theta \leq u_{ij} \text{ and integer}\}$. The function $H_{ij}(x_{ij})$ is the lower envelope of the $(u_{ij} - l_{ij} + 1)$ linear functions of x_{ij} , and hence is a piecewise linear concave function. Using Property 1 it can be shown that each line segment $F_{ij}(\theta) - \theta x_{ij}$ is represented in the lower envelope $H_{ij}(x_{ij})$, and it starts at $b_{ij}(\theta - 1)$ and ends at $b_{ij}(\theta)$. ♦

We have so far transformed the Lagrangian multiplier problem to maximizing a concave cost flow problem. We can alternatively restate this problem as

$$\text{Minimize } \sum_{(i,j) \in A} C_{ij}(x_{ij}) \quad (10a)$$

subject to

$$\sum_{\{j:(j,i) \in A\}} x_{ji} - \sum_{\{j:(i,j) \in A\}} x_{ij} = 0 \quad \text{for all } i \in N, \quad (10b)$$

where $C_{ij}(x_{ij}) = -H_{ij}(x_{ij})$. Hence slopes of the linear segments in $C_{ij}(x_{ij})$ are negative of the corresponding slopes in $H_{ij}(x_{ij})$. Since $H_{ij}(x_{ij})$ is a piecewise linear concave function, $C_{ij}(x_{ij})$ is a piecewise linear convex function. The problem (10) is a convex cost flow problem without lower and upper bounds on arc flows. However, the algorithm described in Section 4 requires lower and upper bounds on arc flows. It can be shown that there exists an optimal solution of (10) such that x_{ij} satisfies $-M \leq x_{ij} \leq M$ for each $(i, j) \in A$, where M is a sufficiently large number. We shall henceforth assume that the flow x also satisfies the following constraint:

$$-M \leq x_{ij} \leq M \quad \text{for all } (i, j) \in A. \quad (10c)$$

4 Cost Scaling Algorithm

The problem (10) is a convex cost flow problem in which the cost associated with the flow on an arc (i, j) is a piecewise linear convex function containing at most U linear segments. It is well known that such problems can be transformed to a minimum cost flow problem by introducing an arc for every linear segment in the cost function (see, for example, Ahuja, Magnanti and Orlin [1993, Chapter 14]). In the case that the number of breakpoints for each cost function is bounded by a constant, one can solve

the resulting minimum cost flow problem efficiently; otherwise this approach yields an exponential-time algorithm. The algorithm by Hochbaum and Shantikumar [1990] solves this problem in time polynomial in terms of the log of the number of breakpoints. In this paper, we develop another algorithm for solving the minimum convex cost network flow problem which has better time complexity.

Our algorithm is an adaptation of the cost scaling algorithm due to Goldberg and Tarjan [1987]. The running time of our algorithm for the dual network flow problem is $O(nm \log(n^2/m) \log(nU))$ which is comparable to the time taken by the Goldberg-Tarjan (GT) algorithm for the minimum cost flow problem. We refer to our algorithm as the *convex cost scaling algorithm* to differentiate it from GT's linear cost scaling algorithm. Our subsequent discussion requires some familiarity with the linear cost scaling algorithm and we refer the readers to the paper by Goldberg and Tarjan [1987] or the book of Ahuja, Magnanti and Orlin [1993] for a description of this algorithm.

The cost scaling algorithm maintains a pseudoflow at each step. A pseudoflow x is any function $x: A \rightarrow \mathbb{R}$ satisfying $-M \leq x_{ij} \leq M$ for all $(i, j) \in A$. For any pseudoflow x , we define the imbalance of node i as

$$e(i) = \sum_{\{j:(i,j) \in A\}} x_{ji} - \sum_{\{j:(i,j) \in A\}} x_{ij} \text{ for all } i \in N. \quad (11)$$

If $e(i) > 0$ for some node i , we refer to node i as an *excess node* and refer to $e(i)$ as the *excess* of node i . We refer to a pseudoflow x with $e(i) = 0$ for all $i \in N$ as a flow. The cost scaling algorithm also maintains a value $\pi(i)$ for each node $i \in N$. We refer to the vector π as a vector of *node potentials*.

The cost scaling algorithm proceeds by constructing and manipulating the residual network $G(x)$ defined as follows with respect to a pseudoflow x . For each $(i, j) \in A$, the residual network $G(x)$ contains two arcs (i, j) and (j, i) . The cost of sending y_{ij} units of flow in arc (i, j) is $C_{ij}(x_{ij} + y_{ij}) - C_{ij}(x_{ij})$; that is, it is the incremental cost of increasing the flow in (i, j) by y_{ij} units. Notice that y_{ij} can be positive as well as negative. We set the cost of arc (i, j) in the residual network as $c_{ij} = C_{ij}(x_{ij} + 1) - C_{ij}(x_{ij})$ and set the cost of arc (j, i) as $c_{ji} = C_{ij}(x_{ij} - 1) - C_{ij}(x_{ij})$. Thus c_{ij} is the marginal cost of increasing the flow in arc (i, j) and c_{ji} is the marginal cost of decreasing flow in arc (i, j) . By Theorem 2, both c_{ij} and c_{ji} are integer valued. The following result is well known for convex cost flows (see for example, Rockafellar [1984]).

Theorem 3. *A flow x is an optimal flow of (10) if and only if $G(x)$ contains no negative cost directed cycle.*

For a given residual network $G(x)$ and a set of node potentials π , we define the *reduced cost* of an arc (i, j) as $c_{ij}^\pi = c_{ij} - \pi(i) + \pi(j)$. Just like its linear counterpart, the convex cost scaling algorithm relies on the concept of approximate optimality. A flow or a pseudoflow x is said to be ε -optimal for some $\varepsilon \geq 0$ if for some node potentials π , the pair (x, π) satisfies the following ε -optimality conditions:

$$c_{ij}^\pi \geq -\varepsilon \text{ for every arc } (i, j) \text{ in } G(x). \quad (12)$$

We call an arc (i, j) *admissible* if $-\varepsilon \leq c_{ij}^\pi < 0$. We shall use the following result in our algorithm. This result uses the fact that each arc in $G(x)$ has an integer cost.

Lemma 1. (Bertsekas [1979]). *If $\varepsilon < 1/n$, then any ε -optimal feasible flow of (10) is an optimal flow.*

We next define the residual capacity r_{ij} of an arc (i, j) in $G(x)$ with respect to a flow x and a set of node potential π . The residual capacity r_{ij} of an arc (i, j) in $G(x)$ is the additional flow we need to send on the arc (i, j) so that the reduced costs of both the arcs (i, j) and (j, i) become nonnegative, that is, if one sends r_{ij} units of flow in (i, j) and then recomputes the residual network, then $c_{ij}^\pi \geq 0$ and $c_{ji}^\pi \geq 0$. Alternatively, the residual capacity r_{ij} of an arc (i, j) in $G(x)$ is the flow y_{ij} that minimizes $(C_{ij}(x_{ij} + y_{ij}) : y_{ij} \geq 0)$. The following lemma obtains a formula for r_{ij} .

Lemma 2. *Let $\theta = \lfloor \pi(i) - \pi(j) \rfloor$. Suppose that (i, j) is an arc with $c_{ij}^\pi < 0$. Then, $r_{ij} = b_{ij}(\theta) - x_{ij}$, if $\theta \leq u_{ij} - 1$; otherwise $r_{ij} = M - x_{ij}$.*

Proof Sketch: If $\theta \leq u_{ij} - 1$, then sending r_{ij} units of flow makes the flow on this arc equal to $b_{ij}(\theta)$. Using Theorem 2, it can be shown that both the reduced costs c_{ij}^π and c_{ji}^π are nonnegative. If $\theta \leq u_{ij} - 1$, then we augment $M - x_{ij}$ units of flow, which eliminates the arc (i, j) from the residual network and makes c_{ji}^π nonnegative. ♦

The convex cost scaling algorithm treats ε as a parameter and iteratively obtains ε -optimal flows for successively smaller values of ε . Initially, $\varepsilon = U$. The algorithm performs cost scaling phases by repeatedly applying an *improve-approximation* procedure that transforms an ε -optimal flow into an $\varepsilon/2$ -optimal flow. After $\lceil \log(nU) \rceil + 1$ scaling phases, $\varepsilon < 1/n$ and the algorithm terminates with an optimal flow. Figure 1 gives the algorithm description.

algorithm *cost scaling*;

begin

$\pi := 0; x := 0; \varepsilon := U;$

while $\varepsilon \geq 1/n$ **do**

begin

 improve-approximation(ε, x, π);

$\varepsilon := \varepsilon/2;$

end;

x is an optimal flow for (10);

end;

(a)

```

procedure improve-approximation;
begin
  for each arc  $(i, j)$  in  $G(x)$  do
    if  $c_{ij}^\pi < 0$  then send  $r_{ij}$  amount of flow on arc  $(i, j)$ ;
  while the network contains an excess node do
    begin
      select an excess node  $i$ ;
      if  $G(x)$  contains an admissible arc  $(i, j)$  then
        push  $\delta := \min\{e(i), r_{ij}\}$  units of flow from node  $i$  to node  $j$ 
      else  $\pi(i) := \pi(i) + \varepsilon/2$ ;
    end;
end;

```

(b)

Fig. 1. The convex cost scaling algorithm.

In the algorithm, we call the operation of sending $\delta = \min\{e(i), r_{ij}\}$ units of flow over the admissible arc (i, j) a *push*. If $\delta = r_{ij}$, we refer to the push as *saturating* and *nonsaturating* otherwise. We also refer to the operation of increasing the potential of node i from $\pi(i)$ to $\pi(i) + \varepsilon/2$ as a *relabel* operation. The purpose of a relabel operation is to create new admissible arcs emanating from this node.

Goldberg and Tarjan had observed that their linear cost scaling algorithm could be extended to treat convex cost flows; moreover, the bounds on the number of pushes per scaling phase were the same as for the linear cost flow problem. Tarjan [1998] communicated this to one of the co-authors of this paper. The algorithm gives an ε -optimal flow at the end of the ε -scaling phase. However, for the general convex cost functions, the algorithm is not guaranteed to give an optimal flow no matter how small ε is. But in our case, the cost functions have integer slopes, and the algorithm gives an optimal flow if $\varepsilon < 1$. The running time analysis of the cost scaling algorithm for the convex cost case does not appear in the published literature; hence we give a brief sketch of this analysis here.

Theorem 4. *The convex cost scaling algorithm correctly solves the dual network flow problem in $O(nm \log(n^2/m) \log(nU))$ time.*

Proof Sketch: The running time of $O(nm \log(n^2/m) \log(nU))$ for the linear cost scaling algorithm relies on the following two critical facts: (i) in a scaling phase any node is relabeled $O(n)$ times; and (ii) in between two consecutive saturating pushes of any arc (i, j) node i must be relabeled at least once. These two facts hold for the convex cost scaling algorithm too. The bound on the number of non-saturating pushes also works as before. The dynamic tree data structure also works as it does for the linear cost case. All we need for it to work is that the residual capacities should behave well in the following sense: if the residual capacity is r_{ij} , and if you send k units of flow in the arc, then the residual capacity reduces to $r_{ij} - k$. This result holds from the manner in which we define the residual capacities. Hence the convex cost

scaling algorithm runs in $O(nm \log(n^2/m))$ time per scaling phase and in $O(nm \log(n^2/m) \log(nU))$ time overall. ♦

The cost scaling algorithm upon termination gives an optimal flow x^* and the optimal node potentials π^* . Both the x^* and π^* may be non-integer. Since the objective function in the convex cost flow problem (10) is piecewise linear, it follows that there always exist optimal node potentials π . To determine those, we construct $G(x^*)$ and solve a shortest path problem to determine shortest path distance $d(i)$ from node 0 to every other node $i \in N$. Then $\pi(i) = -d(i)$ for each $i \in N$ gives an integer optimal set of node potentials for the problem (12). Now recall that $C_{ij}(x_{ij}) = -H_{ij}(x_{ij})$ for each $(i, j) \in A$. This implies that $\mu(i) = -\pi(i) = d(i)$ for each $i \in N$ gives optimal dual variables for (8) and these $\mu(i)$ together $w_{ij} = \mu(i) - u(j)$ for each $(i, j) \in A$ give an optimal solution of the dual network flow problem (2).

5 Generalizations of the Dual Network Flow Problem

In our formulation of the dual network flow problem we have assumed that the constraints $\mu_i - \mu_j = w_{ij}$ are in the equality form. We will show in this section that the constraints of the forms $\mu_i - \mu_j \leq w_{ij}$ can be transformed to the equality form; hence there is no loss of generality by restricting the constraints to the equality form. Suppose that we wish to solve the following problem:

$$\text{Minimize } \sum_{(i,j) \in Q} F_{ij}(w_{ij}) + \sum_{i \in P} B_i(\mu_i) \quad (13a)$$

subject to

$$\mu_i - \mu_j \leq w_{ij} \quad \text{for all } (i, j) \in Q, \quad (13b)$$

$$l_{ij} \leq w_{ij} \leq u_{ij} \quad \text{for all } (i, j) \in Q, \quad (13c)$$

$$l_i \leq \mu_i \leq u_i \quad \text{for all } i \in P. \quad (13d)$$

Let w_{ij}^* denote the value of w_{ij} for which $F_{ij}(w_{ij})$ is minimum. In case there are multiple values for which $F_{ij}(w_{ij})$ is minimum, choose the minimum such value. Let us define the function $E_{ij}(w_{ij})$ in the following manner:

$$E_{ij}(w_{ij}) = \begin{cases} F_{ij}(w_{ij}^*) & \text{if } w_{ij} \leq w_{ij}^* \\ F_{ij}(w_{ij}) & \text{if } w_{ij} > w_{ij}^* \end{cases} \quad (14)$$

Now consider the following problem:

$$\text{Minimize } \sum_{(i,j) \in Q} E_{ij}(w_{ij}) + \sum_{i \in P} B_i(\mu_i) \quad (15a)$$

subject to

$$\mu_i - \mu_j = w_{ij} \quad \text{for all } (i, j) \in Q, \quad (15b)$$

$$l_{ij} \leq w_{ij} \leq u_{ij} \quad \text{for all } (i, j) \in Q, \quad (15c)$$

$$l_i \leq \mu_i \leq u_i \quad \text{for all } i \in P. \quad (15d)$$

The following lemma establishes a relationship between optimal solutions of (13) and (15).

Lemma 3. *For every optimal solution $(\bar{w}, \bar{\mu})$ of (13), there is an optimal solution $(\hat{w}, \hat{\mu})$ of (15) of the same cost, and the converse also holds.*

Proof Sketch: Consider an optimal solution $(\bar{w}, \bar{\mu})$ of (13). We can show that the solution $(\hat{w}, \hat{\mu})$ of with $\hat{w}_{ij} = \min\{\bar{\mu}_i - \bar{\mu}_j, \bar{w}_{ij}\}$ is an optimal solution of (15) with the same cost. Similarly, it can be shown that if $(\hat{w}, \hat{\mu})$ is an optimal solution of (15), then the solution $(\bar{w}, \hat{\mu})$ constructed as $\bar{w}_{ij} = \max\{\hat{\mu}_i - \hat{\mu}_j, \hat{w}_{ij}\}$ is an optimal solution of (13). ♦

6 Applications of the Dual Network Flow Problem

The dual network flow problem and its special cases arise in many application settings. Roundy [1986] formulates a lot-sizing problem in a multi-product, multi-stage, production/inventory system as a dual network flow problem. Boros and Shamir [1991] describe an application of the dual network flow problem in solving a quadratic cost machine scheduling problem. Several multi-facility location problems have constraints of the form (1b)–(1d) (see for example, Ahuja Magnanti and Orlin [1993, Chapter 19]). The convex cost version of these location problems will be dual network flow problems. We describe next in detail three applications of dual network flow problems, some of which we have encountered in our previous research.

Application 1: Inverse Spanning Tree Problem with Convex Costs

Consider an undirected network $G = (N, A)$ with the node set N and the arc set A . Let $n = |N|$ and $m = |A|$. We assume that $N = \{1, 2, \dots, n\}$ and $A = \{a_1, a_2, \dots, a_m\}$. Let c_j denote the cost of the arc a_j . In the inverse spanning tree problem we are given a spanning tree T^0 of G which may or may not be a minimum spanning tree of G and we wish to perturb the arc cost vector c to d so that T^0 becomes a minimum spanning tree with d as the cost vector and $\sum_{j=1}^n F_j(d_j - c_j)$ is minimum, where each $F_j(d_j - c_j)$ is a convex function of d_j . Sokkalingam, Ahuja and Orlin [1999], and Ahuja and Orlin [1998] have studied special cases of the inverse spanning tree problem with cost functions as $\sum_{j=1}^n F_j|d_j - c_j|$ and $\max\{|d_j - c_j| : 1 \leq j \leq m\}$.

We assume without any loss of generality that $T^0 = \{a_1, a_2, \dots, a_{n-1}\}$. We refer to the arcs in T^0 as *tree arcs* and the arcs not in T^0 as *nontree arcs*. In the given spanning tree T^0 , there is a unique path between any two nodes; we denote by $W[a_j]$ the set of tree arcs contained between the two endpoints of the arc a_j . It is well known (see, for example, Ahuja, Magnanti and Orlin [1993]) that T^0 is a minimum spanning tree with respect to the arc cost vector d if and only if

$$d_i \leq d_j \text{ for each } a_i \in W[a_j] \text{ and for each } j = n, n+1, \dots, m. \quad (17)$$

We can convert the inequalities in (17) into equations by introducing nonnegative slack variables w_{ij} 's. Let $P = \{1, 2, \dots, m\}$ and $Q = \{(i, j) : a_i \in T^0 \text{ and } a_j \in W[a_i]\}$. Then, in this notation, the inverse spanning tree problem can be formulated as the following optimization problem:

$$\text{Minimize } \sum_{i \in P} F_i(d_i - c_i) \quad (18a)$$

subject to

$$d_i - d_j = w_{ij} \quad \text{for each } (i, j) \in Q, \quad (18b)$$

$$w_{ij} \geq 0 \quad \text{for each } (i, j) \in Q. \quad (18c)$$

This problem is clearly an instance of the dual network flow problem.

Application 2: Time-Cost Tradeoff Problem in Project Scheduling

Project scheduling is an important management tool used in numerous industries regularly. We will show that the *time-cost tradeoff problem*, an important problem in project scheduling, can be formulated as a dual network flow problem. We refer the reader to the book by Elmaghraby [1978] for a comprehensive discussion of the applications of network models in project scheduling.

We can envision a project as a directed graph $G = (N, A)$ where the arc set A represents the jobs of the project and the node set N represents *events*, denoting the beginning and ending of jobs. Different jobs in the network have precedence relations. We require that all jobs directed into any node must be completed before any job directed out of the node begins. We let node s designate the beginning of the project and node T designate the ending of the project.

Normally, in project scheduling we assume that the job completion time is fixed; however, we here consider a more general settings where job completion times are variable. Let t_{ij} denote the completion time of job (i, j) . We allow t_{ij} to vary in the range $[\alpha_{ij}, \beta_{ij}]$ and associate a convex cost function $F_{ij}(t_{ij})$ which captures the cost of completing the job for different completion times in the range $[\alpha_{ij}, \beta_{ij}]$. We also have another set of decision variables v_i 's denoting *event times*; the variable v_i denotes the time when jobs emanating from node i can begin. The time-cost tradeoff problem is to determine the minimum cost of completing the project in a specified time period T . This problem can be mathematically stated as follows:

$$\text{Minimize } \sum_{(i,j) \in A} F_{ij}(t_{ij}) \quad (19a)$$

subject to

$$v_t - v_s \leq T \quad (19b)$$

$$v_j - v_i \geq t_{ij} \quad \text{for all } (i, j) \in A, \quad (19c)$$

$$\alpha_{ij} \leq t_{ij} \leq \beta_{ij} \quad \text{for all } (i, j) \in A. \quad (19d)$$

Observe that the constraints (19b) capture the fact that the project must be completed within the time period T , and the constraints (19c) imply that for every arc $(i, j) \in A$, event j can occur only after w_{ij} time later than the occurrence of event i . The formulation (19) is an instance of the dual network flow problem.

Application 3: Isotonic Regression Problem

The *isotonic regression problem* can be defined as follows. Given a set $A = \{a_1, a_2, \dots, a_n\} \in \mathbb{R}^n$, find $X = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^n$ so as to

$$\text{Minimize } \sum_{j=1}^n B_j(x_j - a_j) \quad (21a)$$

subject to

$$x_j \leq x_{j+1} \quad \text{for all } j = 1, 2, \dots, n-1, \quad (21b)$$

$$l_j \leq x_j \leq u_j \quad \text{for all } j = 1, 2, \dots, n-1, \quad (21c)$$

where $B_j(x_j - a_j)$ is a convex function for every j , $1 \leq j \leq n$. The isotonic regression problem arises in statistics, production planning, and inventory control (see for example, Barlow et al. [1972] and Robertson et al. [1988]). As an application of the isotonic regression, consider a fuel tank where fuel is being consumed at a slow rate and measurements of the fuel tank are being taken at different points in time. Suppose these measurements are $a_1, a_2, \dots, a_n$. Due to the errors in the measurements, these numbers may not be in the non-increasing order despite the fact that the true amounts of fuel remaining in the tank are non-increasing. However, we need to determine these measurements as accurately as possible. One possible way to accomplish this could be to perturb these numbers to $x_1 \geq x_2 \geq \dots \geq x_n$ so that the cost of perturbation given by $\sum_{j=1}^n B_j(x_j - a_j)$ is minimum, where each $B_j(x_j - a_j)$ is a convex function. We can transform this problem to the isotonic regression problem by replacing x_j 's by their negatives.

If we define $P = \{1, 2, \dots, n\}$ and $Q = \{(j, j+1) : j = 1, 2, \dots, n-1\}$ and require that x_j must be integer, then the isotonic regression problem can be cast as a dual network flow problem. However, it is a very special case of the dual network flow problem and more efficient algorithms can be developed to solve it compared to the dual network flow problem. Ahuja and Orlin [1998] recently developed an $O(n \log U)$ algorithm to solve the isotonic regression problem.

Reference

- Ahuja, R. K., T. L. Magnanti, and J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, NJ.
- Ahuja, R. K., and J. B. Orlin. 1997. A fast algorithm for the bipartite node weighted matching on path graphs with application to the inverse spanning tree problem. Working Paper, Sloan School of Management, MIT, Cambridge, MA.
- Barlow, R. E., D. J. Bartholomew, D. J. Bremner, and H. D. Brunk. 1972. *Statistical Inference under Order Restrictions*. Wiley, New York.
- Bertsekas, D. P. 1979. A distributed algorithm for the assignment problem. Working Paper, Laboratory for Information and Decision Sciences, MIT, Cambridge, MA.
- Boros, E., and R. Shamir. 1991. Convex separable minimization over partial order constraints. Report No. RRR 27-91, RUTCOR, Rutgers University, New Brunswick, NJ.

- Elmaghraby, S. E. 1978. *Activity Networks: Project Planning and Control by Network Models*. Wiley-Interscience, New York.
- Hochbaum, D.S., and J.G. Shanthikumar. 1990. Convex separable optimization is not much harder than linear optimization. *Journal of ACM* 37, 843-862.
- Karzanov, A. V., and S. T. McCormick. 1997. Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM Journal on Computing* 4, 1245-1275.
- McCormick, S. T. 1998. Personal communications.
- Robertson, T., F. T. Wright, and R. L. Dykstra. 1988. *Order Restricted Statistical Inference*. John Wiley & Sons, New York.
- Rockafellar, R. T. 1984. *Network Flows and Monotropic Optimization*. Wiley-Interscience, New York.
- Roundy, R. 1986. A 98% effective lot-sizing rule for a multi-product, multi-stage, production-/inventory system. *Mathematics of Operations Research* 11, 699-727.
- Sokkalingam, P. T., R. K. Ahuja, and J. B. Orlin. 1999. Solving inverse spanning tree problems through network flow techniques. *Operations Research*, March-April Issue.
- Tarjan, R. E. 1998. Personal Communications.

Some Structural and Algorithmic Properties of the Maximum Feasible Subsystem Problem

Edoardo Amaldi¹, Marc E. Pfetsch², and Leslie E. Trotter, Jr.¹

¹ School of OR&IE, Rhodes Hall, Cornell University, Ithaca, NY 14853, USA
{amaldi, ltrotter}@cs.cornell.edu

² Department of Mathematics, MA 7-1, TU Berlin, Germany
pfetsch@math.tu-berlin.de

Abstract. We consider the problem MAX FS: For a given infeasible linear system, determine a *largest feasible subsystem*. This problem has interesting applications in linear programming as well as in fields such as machine learning and statistical discriminant analysis. MAX FS is NP-hard and also difficult to approximate. In this paper we examine structural and algorithmic properties of MAX FS and of *irreducible infeasible subsystems* (IISs), which are intrinsically related, since one must delete at least one constraint from each IIS to attain feasibility. In particular, we establish: (i) that finding a smallest cardinality IIS is NP-hard as well as very difficult to approximate; (ii) a new *simplex decomposition* characterization of IISs; (iii) that for a given clutter, realizability as the IIS family for an infeasible linear system subsumes the Steinitz problem for polytopes; (iv) some results on the *feasible subsystem polytope* whose vertices are incidence vectors of feasible subsystems of a given infeasible system.

1 Introduction

We consider the following combinatorial problem related to infeasible linear inequality systems.

Max FS: *Given an infeasible system $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ with $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, find a feasible subsystem containing as many inequalities as possible.*

This problem has several interesting applications in various fields such as statistical discriminant analysis, machine learning and linear programming (see [2, 26, 22] and the references therein). In the latter case, it arises when the LP formulation phase yields infeasible models and one wishes to diagnose and resolve infeasibility by deleting as few constraints as possible, which is the complementary version of MAX FS [19, 27, 12]. In most situations this cannot be done by inspection and the need for effective algorithmic tools has become more acute with the considerable increase in model size. In fact, MAX FS turns out to be NP-hard [10] and it does not admit a polynomial time approximation scheme unless $P = NP$ [3]. The above complementary version, in which the goal is to delete as few inequalities as possible in order to achieve feasibility, is equivalent to solve to optimality, but is much harder to approximate than MAX FS [5, 4].

Not surprisingly, minimal infeasible subsystems, first discussed in Motzkin's thesis [25], play a key role in the study of MAX FS. A subsystem Σ' of Σ is an *irreducible infeasible subsystem* (IIS) when Σ' is infeasible, but every proper subsystem of Σ' is feasible. In order to help the modeler resolve infeasibility of large linear inequality systems, attention was first devoted to the problem of identifying IISs with a small and possibly minimum number of inequalities [19]; see [14, 13] for several heuristics, now available in commercial solvers such as CPLEX and MINOS [11]. Clearly, when there are many overlapping IISs, this does not provide enough information to repair the original system. To achieve feasibility, one must delete an inequality from each IIS. If all IISs were known, the complementary version of MAX FS could be formulated as the following covering problem [17].

Min IIS Cover: *Given an infeasible system $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ with $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ and the set $\mathcal{C}$ of all its IISs, minimize $\sum_{i=1}^m y_i$ subject to $\sum_{i \in C} y_i \geq 1$ $\forall C \in \mathcal{C}$, $y_i \in \{0, 1\}$, $1 \leq i \leq m$.*

Note that $|\mathcal{C}|$ can grow exponentially with m and n [10].

An exact algorithm based on a partial cover formulation is proposed in [26, 27] and heuristics are described in [22, 12]; a collection of infeasible LPs is maintained in the Netlib library. In [29, 30] the class of hypergraphs representing the IISs of infeasible systems is studied and it is shown that in some special cases MAX FS and MIN IIS COVER can be solved in polynomial time in the number of IISs.

In this paper we investigate some structural and algorithmic properties of IISs and of the polytope defined by the convex hull of incidence vectors of feasible subsystems of a given infeasible system. It is worth noting that, although MAX FS with 0 – 1 variables can be easily shown to admit as a special case the graphical problem of finding a maximum independent node set, it has a different structure when the variables are real-valued. Recent work on problems related to MAX FS and IISs includes, for instance, Håstad's breakthrough [20] which bridges the approximability gap for MAX FS on $GF(p)$, as well as the investigation of the problems of determining minimum or minimal witnesses of infeasibility in network flows [1].

Below we denote the i th row of the matrix $A \in \mathbb{R}^{m \times n}$ by $\mathbf{a}^i \in \mathbb{R}^n$, $1 \leq i \leq m$; for $S \subseteq [m] := \{1, \dots, m\}$, A_S denotes the $|S| \times n$ matrix consisting of the rows of A indexed by S . By identifying the i th inequality of the system Σ (i.e., $\mathbf{a}^i \mathbf{x} \leq b_i$) with index i itself, $[m]$ may also refer to Σ .

2 Irreducible Infeasible Subsystems

First we briefly recall the main known structural results regarding IISs. For notational simplicity, we use the same A and $\mathbf{b}$, with $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, to denote either the original system Σ or one of its IISs.

The known characterizations of IISs are based on the following version of the Farkas Lemma. *For any system $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$, either $A\mathbf{x} \leq \mathbf{b}$ is feasible or $\exists \mathbf{y} \in \mathbb{R}^m$, $\mathbf{y} \geq \mathbf{0}$, such that $\mathbf{y}A = \mathbf{0}$ and $\mathbf{y}\mathbf{b} < 0$, but not both.*

Theorem 1 (Motzkin [25], Fan [16]). *The system $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ with $A, \mathbf{b}$ as above is an IIS if and only if $\text{rank}(A) = m - 1$ and $\exists \mathbf{y} \in \mathbb{R}^m, \mathbf{y} > \mathbf{0}$, such that $\mathbf{y}A = \mathbf{0}$ and $\mathbf{y}\mathbf{b} < 0$.*

The rank condition obviously implies that $m \leq n + 1$.

Now let $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ be an infeasible system which is not necessarily an IIS. The following result relates the IISs of Σ to the vertices of a given *alternative polyhedron*. Recall that the *support* of a vector is the set of indices of its nonzero components.

Theorem 2 (Gleeson and Ryan [17]). *Let $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ be an infeasible system with $A, \mathbf{b}$ as above. Then the IISs of Σ are in one-to-one correspondence with the supports of the vertices of the polyhedron*

$$P := \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y}A = \mathbf{0}, \mathbf{y}\mathbf{b} \leq -1, \mathbf{y} \geq \mathbf{0}\}.$$

The inequality in the alternative system can obviously be replaced by the equation $\mathbf{y}\mathbf{b} = -1$. Note that, by using the transformation into Karmarkar's standard form, any polytope can be expressed as $\{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y}A = \mathbf{0}, \mathbf{y}\mathbf{1} = 1, \mathbf{y} \geq \mathbf{0}\}$ for an appropriate matrix A . Theorem 2 can also be stated in terms of rays [27] and elementary vectors [18].

Definition 1. *An elementary vector of a subspace $L \subseteq \mathbb{R}^m$ is a nonzero vector $\mathbf{y}$ that has a minimal number of nonzero components (when expressed with respect to the standard basis of $\mathbb{R}^m$). In other words, if $\mathbf{x} \in L$ and $\text{supp}(\mathbf{x}) \subset \text{supp}(\mathbf{y})$ then $\mathbf{x} = \mathbf{0}$, where $\text{supp}(\mathbf{y})$ denotes the support of $\mathbf{y}$.*

Corollary 1 (Greenberg [18]). *Let $\Sigma : A\mathbf{x} \leq \mathbf{b}$, $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ be an infeasible system. Then $S \subseteq [m]$ corresponds to an IIS of Σ if and only if there exists an elementary vector $\mathbf{y}$ in the subspace $L := \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y}A = \mathbf{0}\}$ with $\mathbf{y}\mathbf{b} < 0$, $\mathbf{y} \geq \mathbf{0}$ such that $S = \text{supp}(\mathbf{y})$.*

The following result establishes an interesting geometric property of the polyhedra obtained by deleting any inequality from an IIS.

Theorem 3 (Motzkin [25]). *Let $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ be an IIS and let $\sigma \in \Sigma$ be an arbitrary inequality of Σ . Then the polyhedron corresponding to $\Sigma \setminus \sigma$, i.e., the subsystem obtained by removal of σ , is an affine convex cone.*

2.1 Minimum Cardinality IISs

We now determine the complexity status of the following problem for which heuristics have been proposed in [14, 13, 26, 27].

Min IIS: *Given an infeasible system $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ as above, find a minimum cardinality IIS.*

To settle the issue left open in [19, 14, 27], we prove that MIN IIS is not only NP -hard to solve optimally but also hard to approximate. Note that, where $DTIME(T(m))$ denotes the class of problems solvable in time $T(m)$, the assumption $NP \not\subseteq DTIME(m^{\text{polylog } m})$ is stronger than $NP \not\subseteq P$, but it is also believed to be extremely unlikely. Results that hold under such an assumption are often referred to as *almost NP -hard*.

Theorem 4. *Assuming $P \neq NP$, no polynomial time algorithm is guaranteed to yield an IIS whose cardinality is at most c times larger than the minimum one, for any constant $c \geq 1$. Assuming $NP \not\subseteq DTIME(m^{\text{polylog } m})$, MIN IIS cannot be approximated within a factor $2^{\log^{1-\varepsilon} m}$, for any $\varepsilon > 0$, where m is the number of inequalities.*

Proof. We proceed by reduction from the following problem: Given a feasible linear system $D\mathbf{z} = \mathbf{d}$, with $D \in \mathbb{R}^{m' \times n'}$ and $\mathbf{d} \in \mathbb{R}^{m'}$, find a solution $\mathbf{z}$ satisfying all equations with as few nonzero components as possible. In [4] we establish that it is (almost) NP -hard to approximate this problem within the same type of factors, but with m replaced by n , the number of variables. Note that the above nonconstant factor grows faster than any polylogarithmic function, but slower than any polynomial one.

For each instance of the latter problem with an optimal solution containing s nonzero components, we construct a particular instance of MIN IIS with a minimum cardinality IIS containing $s+1$ inequalities. Given any instance $(D, \mathbf{d})$, consider the system

$$(D - D - \mathbf{d}) \begin{pmatrix} \mathbf{z}^+ \\ \mathbf{z}^- \\ z_0 \end{pmatrix} = \mathbf{0}, \quad (\mathbf{0}^t \mathbf{0}^t - 1) \begin{pmatrix} \mathbf{z}^+ \\ \mathbf{z}^- \\ z_0 \end{pmatrix} < \mathbf{0}, \quad \mathbf{z}^+, \mathbf{z}^- \geq \mathbf{0}, \quad z_0 \geq 0. \quad (1)$$

Since the strict inequality implies $z_0 > 0$, the system $D\mathbf{z} = \mathbf{d}$ has a solution with s nonzero components if and only if (1) has one with $s+1$ nonzero components. Now, applying Corollary 1, (1) has such a solution if and only if the system

$$\begin{pmatrix} D^t \\ -D^t \\ -\mathbf{d}^t \end{pmatrix} \mathbf{x} \leq \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -1 \end{pmatrix} \quad (2)$$

has an IIS of cardinality $s+1$. Since (2) is the alternative system of (1), the Farkas Lemma implies that exactly one of these is feasible; as (1) is feasible, (2) must be infeasible. Thus (2) is a particular instance of MIN IIS with $m = 2n' + 1$ inequalities in $n = m'$ variables.

Given that the polynomial time reduction preserves the objective function modulo an additive unit constant, we obtain the same type of non-approximability factors for MIN IIS. $\square$

Note that for the similar (but not directly related) problem of determining minimum witnesses of infeasibility in network flows, NP -hardness is established in [1].

2.2 IIS Simplex Decomposition

Here we provide a new geometric characterization of IISs. For $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, let $A^i := A_{[m] \setminus \{i\}}$ and $\mathbf{b}^i := \mathbf{b}_{[m] \setminus \{i\}}$ denote the $(m-1) \times n$ submatrix and, respectively, the $(m-1)$ -dimensional vector obtained by removing the i th row of A and i th component of $\mathbf{b}$. The following result strengthens the initial part of Theorem 1.

Lemma 1. *For any IIS $\{A\mathbf{x} \leq \mathbf{b}\}$, A^i has linearly independent rows, $\forall i$; i.e., $\text{rank}(A^i) = m-1$.*

Proof. According to Theorem 1, there exists a $\mathbf{y} > \mathbf{0}$ such that $\mathbf{y}A = \mathbf{0}$ and $\mathbf{y}\mathbf{b} = -1$ (by scaling $\mathbf{y}\mathbf{b} < 0$). Suppose some proper subset of rows is linearly dependent; i.e. $\exists \mathbf{z}$, such that $\mathbf{z}A = \mathbf{0}$, $\mathbf{z}\mathbf{b} \geq 0$ (without loss of generality) and some $z_k = 0$.

If some $z_i > 0$, consider $(\mathbf{y} - \epsilon \mathbf{z})A = \mathbf{0}$, $(\mathbf{y} - \epsilon \mathbf{z})\mathbf{b} \leq -1$, where $\epsilon = \min\{y_i/z_i > 0 \mid 1 \leq i \leq m, z_i > 0\}$ (and $\mathbf{y}$ is as above). Then $\mathbf{y} - \epsilon \mathbf{z} \geq \mathbf{0}$, the i th component of $\mathbf{y} - \epsilon \mathbf{z}$ is 0 and the Farkas Lemma contradicts minimality of the system $(\mathbf{y} - \epsilon \mathbf{z})$ fulfills the requirements).

If all $z_i \leq 0$, then $-\mathbf{z} \geq \mathbf{0}$, $-\mathbf{z}A = \mathbf{0}$ and $-\mathbf{z}\mathbf{b} \leq 0$; so setting $\mathbf{y} = -\mathbf{z}$ in the Farkas Lemma leads to a contradiction of minimality, provided $-\mathbf{z}\mathbf{b} < 0$. If $-\mathbf{z}\mathbf{b} = \mathbf{0}$, then $(\mathbf{y} + \epsilon \mathbf{z})A = \mathbf{0}$, $(\mathbf{y} + \epsilon \mathbf{z})\mathbf{b} = -1$, with $\epsilon = \min\{y_i/(-z_i) \mid 1 \leq i \leq m, -z_i > 0\}$ leads to a contradiction as above. $\square$

It is interesting to note that this lemma together with Theorem 1 imply that an infeasible system $\{A\mathbf{x} \leq \mathbf{b}\}$ is an IIS if and only if $\text{rank}(A^i) = m-1$ for all i , $1 \leq i \leq m$.

We thus have the following *simplex decomposition* result for IISs.

Theorem 5. *The system $\{A\mathbf{x} \leq \mathbf{b}\}$ is an IIS if and only if $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \geq \mathbf{b}\} = L + Q$, where L is the lineality subspace $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{0}\}$ and Q is an $(m-1)$ -simplex with vertices determined by maximal proper subsystems of $\{A\mathbf{x} = \mathbf{b}\}$; namely, each vertex of Q is a solution for a subsystem $\{A^i\mathbf{x} = \mathbf{b}^i\}$, $1 \leq i \leq m$.*

Proof. ($\Rightarrow$) To see feasibility of $\{A\mathbf{x} \geq \mathbf{b}\}$, delete constraint $\mathbf{a}^i\mathbf{x} \geq b_i$ to get the equality system $\{A^i\mathbf{x} = \mathbf{b}^i\}$. By Lemma 1, this system has a solution, say $\mathbf{x}^i$, and we must have $\mathbf{a}^i\mathbf{x}^i > b_i$, else $\mathbf{x}^i$ satisfies $\{A\mathbf{x} \leq \mathbf{b}\}$. Applying the polyhedral resolution theorem, $P := \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \geq \mathbf{b}\} \neq \emptyset$ can be written as $P = K + Q$, where $K = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \geq \mathbf{0}\}$ is its recession cone and $Q \subseteq P$ is a polytope generated by representatives of its minimal nonempty faces.

If $\mathbf{x}$ satisfies $A\mathbf{x} \geq \mathbf{0}$ and $\mathbf{a}^i\mathbf{x} > 0$ for row $\mathbf{a}^i$ then $\mathbf{x}^i - \epsilon \mathbf{x}$ satisfies $A(\mathbf{x}^i - \epsilon \mathbf{x}) \leq \mathbf{b}$ for sufficiently large $\epsilon > 0$ and the original system $\{A\mathbf{x} \leq \mathbf{b}\}$ would be feasible. Therefore we must have that each $\mathbf{a}^i\mathbf{x} = 0$ for $1 \leq i \leq m$, $\mathbf{x} \in K$ and we get that in fact $K = L := \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{0}\}$.

For Q , minimal nonempty faces of P are given by changing a maximal set of inequalities into equalities (all but one relation). Thus the vectors $\mathbf{x}^i$ obtained

by solving $\{A^i \mathbf{x} = \mathbf{b}^i\}$ determine Q ; i.e., $Q = \text{conv}(\{\mathbf{x}^1, \dots, \mathbf{x}^m\})$. For $A \in \mathbb{R}^{m \times n}$, Q is the $(m-1)$ -simplex generated by the m points $\{\mathbf{x}^1, \dots, \mathbf{x}^m\}$. To see that the $\mathbf{x}^i$ generate an $(m-1)$ -simplex, we must only show that they are affinely independent. But if $\mathbf{x}^i$ is affinely dependent on the other $\mathbf{x}^j$, then $\mathbf{x}^i = \sum_{j \neq i} \lambda_j \mathbf{x}^j$ with $\sum_{j \neq i} \lambda_j = 1$. Thus we have $\mathbf{a}^i \mathbf{x}^i > b_i$, but also $\mathbf{a}^i \mathbf{x}^i = \mathbf{a}^i (\sum_{j \neq i} \lambda_j \mathbf{x}^j) = \sum_{j \neq i} \lambda_j (\mathbf{a}^i \mathbf{x}^j) = \sum_{j \neq i} \lambda_j b_i = b_i$, which is a contradiction.

($\Leftarrow$) If the system $\{A\mathbf{x} \leq \mathbf{b}\}$ is infeasible, then the minimality is obvious, because the simplex conditions on Q imply that every proper subsystem has an equality solution.

To show that $\{A\mathbf{x} \leq \mathbf{b}\}$ is infeasible, assume for the sake of contradiction that $\hat{\mathbf{x}} \in \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \leq \mathbf{b}\} \neq \emptyset$ and $\hat{\mathbf{x}}$ satisfies a maximal number of these relations at equality. Let $\mathbf{a}^i \hat{\mathbf{x}} < b_i$ and note that for $\mathbf{x}^i$ defined as above, we have $\mathbf{a}^i \mathbf{x}^i > b_i$. Thus we can set $\lambda = (\mathbf{a}^i \mathbf{x}^i - b_i) / (\mathbf{a}^i \mathbf{x}^i - \mathbf{a}^i \hat{\mathbf{x}})$ and have $0 < \lambda < 1$, so that $\mathbf{a}^i (\lambda \hat{\mathbf{x}} + (1-\lambda)\mathbf{x}^i) = b_i$. But then at $\lambda \hat{\mathbf{x}} + (1-\lambda)\mathbf{x}^i$ more relations of $\{A\mathbf{x} \leq \mathbf{b}\}$ hold at equality than at $\hat{\mathbf{x}}$, contradicting the choice of $\hat{\mathbf{x}}$. $\square$

According to the above proof, we can take the $\mathbf{x}^i$'s as the representatives of the minimal nonempty faces of $\{A\mathbf{x} \leq \mathbf{b}\}$ that lie in $L^\perp$; i.e., $Q \subset L^\perp$. By Lemma 1, we know that $\{\mathbf{x} \in \mathbb{R}^n \mid A^i \mathbf{x} = \mathbf{b}^i\} = \mathbf{x}^i + L$, where L is the lineality space of the original linear system $\{A\mathbf{x} \geq \mathbf{b}\}$.

It is worth observing that Theorem 5 handles the following special cases.

If $m = 1$, then A has only one row, say $\{A\mathbf{x} \leq \mathbf{b}\} = \{\mathbf{0}\mathbf{x} \leq -1\}$. Thus $L = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{0}\mathbf{x} = 0\} = \mathbb{R}^n$ and $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{0}\mathbf{x} \geq -1\} = \mathbb{R}^n + \{0\} = L + Q = L$.

If $m = n + 1$, then A has $n + 1$ rows. Assuming A to be of full column rank, $L = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = 0\} = \{0\}$ and $Q = \text{conv}(\{\mathbf{x}_1, \dots, \mathbf{x}_{n+1}\})$ is an n -simplex and $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \geq \mathbf{b}\} = \{0\} + Q$.

3 IIS-Hypergraphs

Consider for any infeasible system the following hypergraph.

Definition 2. Given an infeasible system $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ with $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, $H = (V, \mathcal{E})$ is the IIS-hypergraph of Σ if

- i. the nodes in V are in one-to-one correspondence with the inequalities of Σ ,
- ii. the hyperedges in $\mathcal{E}$ are in one-to-one correspondence with the IISs of Σ and each hyperedge contains precisely the nodes associated to the inequalities contained in the corresponding IIS.

Investigations on the structure of *IIS-hypergraphs* began with [29, 30]. In particular, it was shown that IIS-hypergraphs do not share many properties with other known classes of hypergraphs generalizing bipartite graphs. Indeed, IIS-hypergraphs (with no trivial IISs of cardinality 1) just turn out to be *bi-colourable*; i.e., their nodes can be partitioned into two subsets so that neither subset contains a hyperedge. Note, however, that there is more structure for

IIS-hypergraphs than simply bicolourability, as there will generally exist many different bipartitions into two feasible subsystems [29, 18].

In IIS-hypergraph terminology, MIN IIS COVER amounts to finding a minimum cardinality *transversal*, i.e., a subset of nodes having nonempty intersection with every hyperedge. The special structure of the IIS-hypergraphs accounts for the fact that the greedy algorithm is guaranteed to find a minimum transversal for those with nondegenerate alternative polyhedra [30] (a subclass of uniform hypergraphs) while the problem is *NP*-hard even for simple graphs, i.e., for 2-uniform hypergraphs.

Here we address the fundamental problem of recognizing IIS-hypergraphs. For the definitions of a poset and (face) lattice see, e.g., [31].

Let E be a finite set and $\mathcal{F}$ a clutter on E . The poset $\mathcal{L}(\mathcal{F}) = (S, \leq)$ can be constructed as follows. $S \subseteq 2^E$ and the relation “ $\leq$ ” on S is the set inclusion. A subset U of E is in S if U is the intersection of elements of $\mathcal{F}$. The element $\hat{1} := \bigcup\{F \in \mathcal{F}\}$ is also in S . Notice that the zero $\hat{0} := \bigcap\{F \in \mathcal{F}\}$ is always in S and is possibly the empty set. Then $\mathcal{L}(\mathcal{F})$ is a lattice with the meet defined by intersection. Note that the size of $\mathcal{L}(\mathcal{F})$ can be exponential in the size of $\mathcal{F}$.

The *face lattice* of a polytope P is its set of faces, ordered by inclusion, with the meet defined by intersection. It is well known (see, e.g., [31]) that the face lattice of P has a rank function $r(\cdot)$ satisfying $r(F) = \dim(F) - 1$ for any face F , and is both atomic and coatomic. Two polytopes with isomorphic face lattices are *combinatorially equivalent*.

Let R denote either $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{A}$ (the real algebraic numbers over $\mathbb{Q}$) or $\mathbb{R}$.

IIS Realizability problem for R : *Given a clutter $\mathcal{C}$ over a finite ground set of cardinality m , does there exist an infeasible linear system $\{A\mathbf{x} \leq \mathbf{b}\}$, with $A \in R^{m \times n}$ and $\mathbf{b} \in R^m$, such that the sets in $\mathcal{C}$ index the IISs of this system?*

In the above definition, infeasibility is meant with respect to $\mathbb{R}$ and n is free. If such a system exists, the clutter $\mathcal{C}$ is *IIS-realizable*. The IIS Realizability problem is obviously equivalent to that of recognizing IIS-hypergraphs. In the sequel we also consider the restricted version of the IIS Realizability problem in which the right-hand side of the linear system is fixed, namely, in which $\mathbf{b} = -\mathbf{1}$.

Steinitz problem for R : *Given a lattice $\mathcal{L}$, does there exist a polytope $P \subset \mathbb{R}^d$ with vertices in R^d such that the face lattice of P is isomorphic to $\mathcal{L}$?*

If the answer is affirmative, $\mathcal{L}$ is *realizable* as a polytope. In this case d can be assumed to be the dimension of $\mathcal{L}$. P can be given either as a (complete) list of vertices or facets. See [9] for related material.

Theorem 6. *The IIS Realizability problem is at least as hard as the Steinitz problem.*

Proof. We show that for any instance of the Steinitz problem we can construct in polynomial time a special instance of the above-mentioned restricted IIS Realizability problem such that the answer to the first instance is affirmative if and only if the answer to the second instance is affirmative. Since face lattices

of polytopes need to be ranked as well as atomic and these properties can be checked in polynomial time, we focus attention on this type of lattices.

Given an arbitrary instance of the Steinitz problem defined by a ranked atomic lattice $\mathcal{L}$, we construct the following special instance of the restricted IIS Realizability problem with $\mathbf{b} = -\mathbf{1}$. Suppose $\mathcal{L}$ contains k atoms and m coatoms. Label arbitrarily the coatoms with the sets $\{1\}, \dots, \{m\}$ and the atoms with the sets $C_1, \dots, C_k$, where C_i includes all the elements in the labels of the coatoms that contain the corresponding atom. Define $\mathcal{C} = \{C_1, \dots, C_k\}$ and $\overline{\mathcal{C}} := \{\overline{C}_1, \dots, \overline{C}_k\}$, where $\overline{C}_i = \{1, \dots, m\} \setminus C_i$. Thus the arbitrary choices of the labeling just correspond to a permutation of coordinates and hence do not change the structure.

If the original instance of the Steinitz problem has a positive answer, there exists a polytope P such that $\mathcal{L}$ is the face lattice of P . According to the remark following Theorem 2, this polytope can be expressed in the special form $P = \{\mathbf{y} \in \mathbb{R}^{m'} \mid \mathbf{y}A = \mathbf{0}, \mathbf{y}\mathbf{1} = 1, \mathbf{y} \geq \mathbf{0}\}$, with $A \in \mathbb{R}^{m' \times n}$ and suitable m', n . Hence $m' > m$, the number of facets, and $\{\mathbf{A}\mathbf{x} \leq -\mathbf{1}\}$ is the infeasible system associated to P .

Since the face lattice of a polytope is coatomic, each face of P can be identified with the set of facets it is contained in. If these sets corresponding to all faces are ordered by set inclusion, one obtains a lattice $\mathcal{L}'$ which is anti-isomorphic to the face lattice of P . The meet is defined by intersection. It is easy to see that the lattice $\mathcal{L}(\mathcal{C})$ is isomorphic to $\mathcal{L}'$. The atoms correspond to the facets of P and the coatoms to its vertices.

By construction, each set C_i (atom of $\mathcal{L}$) corresponds to a vertex $\mathbf{v}^i$ of P . All facets of P are defined by inequalities of the form $y_i \geq 0$. Up to relabeling of the coatoms in the definition of $\mathcal{C}$, the facet defined by $y_i \geq 0$ can be identified with $\{i\}$. Thus $C_i = \{j \in \{1, \dots, m\} \mid v_j^i = 0\}$ and $\overline{C}_i$ is the support of the vertex $\mathbf{v}^i$. By Theorem 2, each $\overline{C}_i$ corresponds to an IIS of the associated infeasible system $\{\mathbf{A}\mathbf{x} \leq -\mathbf{1}\}$ and hence $\overline{\mathcal{C}}$ is IIS-realizable with the restricted type of right-hand side and with a polytope as alternative polyhedron.

Conversely, suppose that the corresponding instance of the restricted IIS Realizability problem with $\mathbf{b} = -\mathbf{1}$ defined by $\overline{\mathcal{C}}$ has a positive answer and consider the alternative polyhedron $P = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y}A = \mathbf{0}, \mathbf{y}\mathbf{1} = 1, \mathbf{y} \geq \mathbf{0}\}$ with $A \in \mathbb{R}^{m \times n}$. As seen above, each $\overline{C}_i$ corresponds to the support of a vertex of P and each C_i corresponds to the set of facets that this vertex lies on, i.e., $\mathcal{L}(\mathcal{C})$ is anti-isomorphic to the face lattice of P . Now the vertex-facet incidence information encoded in $\mathcal{C}$ and the fact that $\mathcal{L}$ is atomic, imply the whole structure of the lattice $\mathcal{L}$. Therefore $\mathcal{L}(\mathcal{C})$ is anti-isomorphic to $\mathcal{L}$ and hence P is a realization of $\mathcal{L}$. $\square$

Given polynomials $f_1, \dots, f_r, g_1, \dots, g_s, h_1, \dots, h_t \in \mathbb{Z}[x_1, \dots, x_l]$, the problem to decide whether the polynomial system $f_1 = \dots = f_r = 0, g_1 \geq 0, \dots, g_s \geq 0, h_1 > 0, \dots, h_t > 0$ has a solution in $R^l = \mathbb{A}^l$ is called the *Existential theory of the reals* (ETR). ETR is polynomial time equivalent to the Steinitz problem for 4-Polytopes over $\mathbb{A}$ [28]. (All polytopes realizable over $\mathbb{R}$ are realizable over

$\mathbb{A}$.) Moreover, ETR is polynomial time equivalent to the Steinitz problem for d -Polytopes with $d + 4$ vertices over $\mathbb{A}$ [24]. Since ETR is easily verified to be NP-hard, the same is valid for the general Steinitz problem (over $\mathbb{A}$) and for the IIS Realizability problem.

According to Theorem 2.7 of [9], for $R = \mathbb{Q}$ or $\mathbb{A}$, to decide whether an arbitrary polynomial $f \in \mathbb{Z}[x_1, \dots, x_l]$ has zeros in R^l , where l is a positive integer, is equivalent to solve the Steinitz problem for R . For $R = \mathbb{Q}$, it is not even clear whether the Steinitz problem (and therefore the IIS Realizability problem) is decidable since finding roots in $R = \mathbb{Q}$ of a single polynomial $f \in \mathbb{Z}[x_1, \dots, x_l]$ is the unsolved rational version of Hilbert's 10th problem. By the well known theorem of Matiyasevic, there does not exist an algorithm for deciding whether f has roots in $\mathbb{Z}$. By the quantifier elimination result of Tarski, the problem is decidable for $R = \mathbb{A}$. Note that, unlike $\mathbb{R}$, $\mathbb{A}$ admits a finite representation. For $R = \mathbb{A}$, it is unknown whether the Steinitz problem is in NP. See [23, 8] and references therein for this and related issues.

4 Feasible Subsystem (FS) Polytope

Consider an infeasible system $\Sigma : \{A\mathbf{x} \leq \mathbf{b}\}$ and let $[m] = \{1, \dots, m\}$ be the set of indices of all inequalities in Σ . If $\mathcal{I}$ denotes the set of all feasible subsystems of Σ , $([m], \mathcal{I})$ is clearly an independence system and its set of circuits $\mathcal{C}(\mathcal{I})$ corresponds to the set of all IISs. We denote by P_{FS} the polytope generated by the convex hull of all the incidence vectors of feasible subsystems.

Let us first briefly recall some definitions and facts about independence system polytopes. To any independence system $(E, \mathcal{I})$ with the family of circuits denoted by $\mathcal{C}(\mathcal{I})$ we can associate the polytope $P(\mathcal{I}) = P(\mathcal{C}(\mathcal{I})) = \text{conv}(\{\mathbf{y} \in \{0, 1\}^{|E|} \mid \mathbf{y} \text{ is the incidence vector of an } I \in \mathcal{I}\})$. The *rank function* is defined by $r(S) = \max\{|I| \mid I \subseteq S, I \in \mathcal{I}\}$ for all $S \subseteq E$. For any $S \subseteq E$, the *rank inequality* for S is $\sum_{e \in S} y_e \leq r(S)$, which is clearly valid for $P(\mathcal{I})$. A subset $S \subseteq E$ is *closed* if $r(S \cup \{t\}) \geq r(S) + 1$ for all $t \in E - S$ and *nonseparable* if $r(S) < r(T) + r(S - T)$ for all $T \subset S$, $T \neq \emptyset$. For any set $S \subseteq E$, S must be closed and nonseparable for the corresponding rank inequality to define a facet of $P(\mathcal{I})$. These conditions generally are only necessary, but sufficient conditions can be stated using the following concept [21]. For $S \subseteq E$, the *critical graph* $G_S(\mathcal{I}) = (S, F)$ is defined as follows: $(e, e') \in F$, for $e, e' \in S$, if and only if there exists an independent set I such that $I \subseteq S$, $|I| = r(S)$ and $e \in I$, $e' \notin I$, $I - e + e' \in \mathcal{I}$. It is shown in [21] that if S is a closed subset of E and the critical graph $G_S(\mathcal{I})$ of $\mathcal{I}$ on S is connected, then the corresponding rank inequality induces a facet of the polytope $P(\mathcal{I})$. See references in [15].

4.1 Rank-Facets of the FS Polytope

As P_{FS} is an independence system polytope, it is full-dimensional if and only if there are no trivially infeasible inequalities in Σ . The inequalities $y_i \geq 0$ are facet defining for all $1 \leq i \leq m$. Moreover, it is easy to verify that for each i the

inequality $y_i \leq 1$ defines a facet of P_{FS} if and only if there is no IIS of cardinality 2 that includes i and P_{FS} is full-dimensional.

In fact, Parker [26] began an investigation of the polytope associated to the MIN IIS COVER problem, considering it as a special case of the general set covering polytope (see references in [15]). Since there is a simple correspondence between set covering polytopes and the complementary independence system polytopes [21], the results in [26] can be translated so that they apply to P_{FS} .

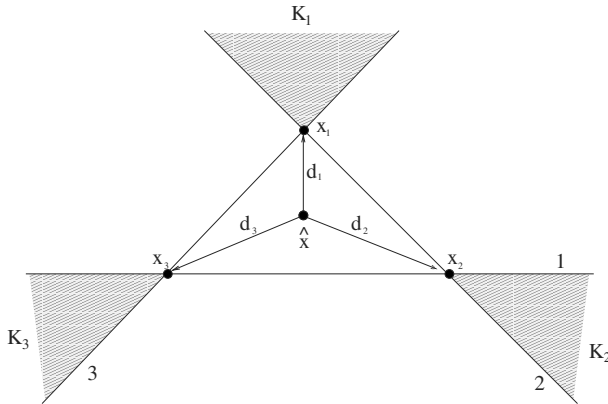
Let S be an arbitrary IIS of Σ , $A_S \mathbf{x} \leq \mathbf{b}_S$ be its corresponding subsystem, and $\sum_{i \in S} y_i \leq r(S) = |S| - 1$ the corresponding (rank) *IIS-inequality*. Since the complementary covering inequality $\sum_{i \in C} y_i \geq 1$ induced by every IIS C is proved to be facet defining in [26], we have:

Theorem 7. *The IIS-inequality arising from any IIS defines a facet of P_{FS} .*

We give here a geometric proof (based on the above-mentioned sufficient conditions [21]), which is simpler than that of [26] and which provides additional insight into the IIS structure.

Proof. It is easy to verify that IIS-inequalities are valid for P_{FS} . Since the critical graph corresponding to any IIS is clearly connected (in fact, a complete graph), we just need to show that every IIS is closed.

a) First consider the case of maximal IISs, i.e. with $|S| = n + 1$.



For each $i \in S$, consider the unique $\mathbf{x}^i = A_{S \setminus \{i\}}^{-1} \mathbf{b}_{S \setminus \{i\}}$. By the proof of Theorem 5, we know that $\mathbf{x}^1, \dots, \mathbf{x}^{n+1}$ are affinely independent. If $\mathbf{d}_i := (\mathbf{x}^i - \hat{\mathbf{x}})$ for all i , $1 \leq i \leq n + 1$, where $\hat{\mathbf{x}} := \frac{1}{n+1} \sum_{i=1}^{n+1} \mathbf{x}^i$, $\mathbf{d}_1, \dots, \mathbf{d}_{n+1}$ are also affinely independent. Clearly $\sum_{i=1}^{n+1} \mathbf{d}_i = \mathbf{0}$ and the $\mathbf{d}_i$'s generate $\mathbb{R}^n$. Since each $\mathbf{x}^i$ satisfies exactly n of the $n + 1$ inequalities in S with equality and $\mathbf{a}^i \mathbf{x}^i > b_i$ (otherwise S would be feasible), we have $\hat{\mathbf{x}} \in \{\mathbf{x} \in \mathbb{R}^n \mid A_S \mathbf{x} \geq \mathbf{b}_S\}$, i.e., $\hat{\mathbf{x}}$ satisfies the reversed inequalities of the IIS. In fact, $\hat{\mathbf{x}}$ is an interior point of the above “reversed” polyhedron.

According to Theorem 3, deleting any inequality from an IIS yields a feasible subsystem that defines an affine cone. For maximal IISs, we have $n + 1$ affine cones $K_i := \mathbf{x}^i + K'_i$, where $K'_i = \{\mathbf{x} \in \mathbb{R}^n \mid A_{S \setminus \{i\}} \mathbf{x} \leq \mathbf{0}\}$ for $1 \leq i \leq n + 1$. Note that the ray generated by $\mathbf{d}_i$ and passing through $\mathbf{x}^i$, i.e., $R_i := \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \mathbf{x}^i + \alpha \mathbf{d}_i, \alpha \geq 0\}$, is contained in K_i because we have

$$A_{S \setminus \{i\}}(\alpha \mathbf{d}_i) = \alpha A_{S \setminus \{i\}}(\mathbf{x}^i - \hat{\mathbf{x}}) = \alpha(\mathbf{b}_{S \setminus \{i\}} - A_{S \setminus \{i\}} \hat{\mathbf{x}}) \leq \mathbf{0},$$

where we used the fact that $A_{S \setminus \{i\}} \hat{\mathbf{x}} \geq \mathbf{b}_{S \setminus \{i\}}$. Now consider an arbitrary inequality $\tilde{\mathbf{a}} \mathbf{x} \leq \tilde{b}$ with $\tilde{\mathbf{a}} \neq \mathbf{0}$. We will verify that $H := \{\mathbf{x} \in \mathbb{R}^n \mid \tilde{\mathbf{a}} \mathbf{x} \leq \tilde{b}\}$ has a nonempty intersection with at least one of the K_i 's, $1 \leq i \leq n + 1$. Thus, for any $t \in E - S$ we have $\text{rank}(S \cup \{t\}) = \text{rank}(S) + 1 = n + 1$, which means that the IIS defined by S is closed.

Since $\mathbf{d}_1, \dots, \mathbf{d}_{n+1}$ generate $\mathbb{R}^n$ and $\sum_{i=1}^{n+1} \mathbf{d}_i = \mathbf{0}$, we have $\sum_{i=1}^{n+1} \tilde{\mathbf{a}} \mathbf{d}_i = \tilde{\mathbf{a}}(\sum_{i=1}^{n+1} \mathbf{d}_i) = 0$ and therefore $\tilde{\mathbf{a}} \neq \mathbf{0}$ implies that we cannot have $\tilde{\mathbf{a}} \mathbf{d}_i = 0 \forall i, 1 \leq i \leq n + 1$. Thus there exists at least one i , such that $\tilde{\mathbf{a}} \mathbf{d}_i < 0$. But this implies that $R_i \cap H \neq \emptyset$. In other words, $K_i \cap H \neq \emptyset$ and this proves the theorem for maximal IISs.

b) The result can be easily extended to non-maximal IISs, i.e., with $|S| < n + 1$. From Theorem 5 we know that $P := \{\mathbf{x} \in \mathbb{R}^n \mid A \mathbf{x} \geq \mathbf{b}\} = L + Q$ with $Q \subseteq L^\perp$. Since P is full-dimensional ($\hat{\mathbf{x}}$ is an interior point), $n = \dim(P) = \dim(L) + \dim(Q)$ and $\dim(Q) = \text{rank}(A_S) = |S| - 1 < n$ imply that $\dim(L) \geq 1$.

Two cases can arise:

- i) If the above-mentioned $\tilde{\mathbf{a}}$ is in $\text{lin}(\{\mathbf{a}^1, \dots, \mathbf{a}^m\}) = L^\perp$, the linear hull of the rows of A , then since $\dim(L^\perp) = \dim(Q)$, we can apply the above result to $L^\perp$.
- ii) If $\tilde{\mathbf{a}} \notin \text{lin}(\{\mathbf{a}^1, \dots, \mathbf{a}^m\}) = L^\perp$, then the projection of $H^\circ := \{\mathbf{x} \in \mathbb{R}^n \mid \tilde{\mathbf{a}} \mathbf{x} = \tilde{b}\}$ onto L yields the whole L and therefore $H = \{\mathbf{x} \in \mathbb{R}^n \mid \tilde{\mathbf{a}} \mathbf{x} \leq \tilde{b}\}$ must have a nonempty intersection with all the cones corresponding to the maximal consistent subsystems of $\{A_S \mathbf{x} \leq \mathbf{b}_S\}$. $\square$

It is worth noting that closedness of every IIS makes P_{FS} quite special among all independence system polyhedra, since the circuits of a general independence system need not be closed.

The **separation problem for IIS-inequalities** is defined as follows: *Given an infeasible system Σ and an arbitrary vector $\mathbf{y} \in \mathbb{R}^m$, show that $\mathbf{y}$ satisfies all IIS-inequalities or find at least one violated by $\mathbf{y}$.*

In view of the trivial valid inequalities, we can assume that $\mathbf{y} \in [0, 1]^m$. Moreover, we may assume with no loss of generality, that the nonzero components of $\mathbf{y}$ correspond to an infeasible subsystem of Σ .

Proposition 1. *The separation problem for IIS-inequalities is NP-hard.*

Proof. We proceed by polynomial time reduction from the decision version of the MIN IIS problem, which is NP-hard according to Theorem 4. Given an infeasible system $\Sigma : \{A \mathbf{x} \leq \mathbf{b}\}$ with m inequalities, n variables and a positive integer K with $1 \leq K \leq n + 1$, does it have an IIS of cardinality at most K ?

Let $(A, \mathbf{b})$ and K define an arbitrary instance of the above decision problem. Consider the particular instance of the separation problem given by the same infeasible system together with the vector $\mathbf{y}$ such that $y_i = 1 - 1/(K+1)$ for all i , $1 \leq i \leq m$.

Suppose that Σ has an IIS of cardinality at most K which is indexed by the set S . Then the corresponding IIS-inequality $\sum_{i \in S} y_i \leq |S| - 1$ is violated by the vector $\mathbf{y}$ because

$$\sum_{i \in S} y_i = \sum_{i \in S} \left(1 - \frac{1}{K+1}\right) = |S| - \frac{|S|}{K+1} > |S| - 1,$$

where the strict inequality is implied by $|S| \leq K$. Thus the vector $\mathbf{y}$ can be separated from P_{FS} .

Conversely, if there exists an IIS-inequality violated by $\mathbf{y}$, then

$$\sum_{i \in S} y_i = |S| - \frac{|S|}{(K+1)} > |S| - 1$$

implies that the cardinality of the IIS defined by S is at most K .

Therefore, the original infeasible system Σ has an IIS of cardinality at most K if and only if some IIS-inequality is violated by the given vector $\mathbf{y}$. $\square$

In [21] the concept of generalized antiwebs, which includes as special cases generalized cliques, generalized odd holes and generalized antiholes, is introduced. Necessary and sufficient conditions are also established for the corresponding rank inequalities to define facets of the associated independence system polytope.

Let m, t, q be integers such that $2 \leq q \leq t \leq m$, let $E = \{e_1, \dots, e_m\}$ be a finite set, and define for each $i \in M = \{1, \dots, m\}$ the subset $E^i = \{e_i, \dots, e_{i+t-1}\}$ (where the indices are taken modulo m) formed by t consecutive elements of E . An (m, t, q) -generalized antiweb on E is the independence system having the following family of subsets of E as circuits:

$$\mathcal{AW}(m, t, q) = \{C \subseteq E \mid C \subseteq E^i \text{ for some } i \in M, |C| = q\}.$$

As mentioned in [21], $\mathcal{AW}(m, t, q)$ corresponds to *generalized cliques* when $m = t$, to *generalized odd holes* when $q = t$ and t does not divide m , and to *generalized antiholes* when $m = qt + 1$. The rank inequality induced by a generalized antiweb $\sum_{i \in E} y_i \leq \lfloor m(q-1)/t \rfloor$ defines a canonical facet of the independence system polytope $P(\mathcal{AW}(m, t, q))$ if and only if $m = t$ or t does not divide $m(q-1)$ [21].

In the case of P_{FS} , the ground set is the set of indices of inequalities in the infeasible system Σ under consideration.

Proposition 2. *No facets of P_{FS} are induced by generalized cliques other than simple IISs (i.e., $m = t = q$).*

Proof. We invoke the following result (Proposition 3.15 of [21]). For any $S \subseteq E$, let $\mathcal{C}_S = \{C \in \mathcal{C} \mid C \subseteq S\}$ denote the family of circuits of the independence system induced by $(E, \mathcal{I})$ on S . Then the rank inequality $\sum_{e \in S} y_e \leq r(S)$ induces a facet of $P(\mathcal{C})$ if and only if S is closed and the rank inequality induces a facet of $P(\mathcal{C}_S)$. Hence it suffices to consider the case $S = E$ and $\mathcal{C}_S = \mathcal{AW}(m, t, q)$.

It is easy to verify that the only (m, t, q) -generalized antiwebs that can arise in IIS-hypergraphs are those with $q = t$. Suppose that $q < t$ and consider E^1 , an arbitrary circuit $C \in \mathcal{AW}(m, t, q)$ with $C \subseteq E^1$ and an arbitrary element $e \in E^1 \setminus C$. By definition of $\mathcal{AW}(m, t, q)$, any q subset of E^1 is a circuit. This must be true in particular for all subsets containing e and $q - 1$ elements of C . But then C cannot be closed because $r(C \cup \{e\}) = r(C)$ and thus we have a contradiction to the fact that all IISs are closed (Theorem 7). Hence the only generalized cliques that can arise are those with $m = t = q$, that is, in which the whole ground set E is an IIS. $\square$

The generalized antiwebs which are not ruled out by the above proof, i.e., $\mathcal{AW}(m, t, q)$ with $q = t$, clearly correspond to simple circular sequences of IISs of cardinality t given by the subsets E^i , $i \in M$, of the definition. For $t = q = 2$, it is easy to see that the only possible cases that can arise as induced hypergraphs of IIS-hypergraphs are those with $m = 4$ and $m = 2$. In fact, we conjecture that no other (m, t, q) -generalized antiwebs can occur besides the cases $m = t = q$ with $q \geq 2$, $m = 4$ and $t = q = 2$ as well as the trivial cases in which $q = 1$. In this respect it is interesting to note that the remark following Theorem 5 implies that the lineality spaces L associated to all the IISs E^i , $i \in M$, in any given generalized antiweb are identical. Therefore we can assume that they are all maximal IISs contained in $L^\perp$ and exploit the special geometric structure of such IISs revealed by the proof of Theorem 7. An intermediate step would then be to show that no sequence of more than 3 such successive IISs E^i can occur without other additional IISs involving t nonsequential elements. In the case $m = 5$ and $t = 2$, this observation is clearly valid.

Besides settling the above-mentioned issue, we are investigating other rank and non-rank facets of P_{FS} . For rank facets, it is also of interest to consider the extent to which the sufficient condition involving connectedness of the critical graph could also be necessary. By enumerating all independence systems on at most 6 elements, we have verified that all cases with rank facets different from IIS-inequalities and with a nonconnected critical graph occur in independence systems which cannot be realized as P_{FS} .

For non-rank facets, we can specialize some known facet classes for general independence system polytopes and set covering polytopes, e.g., the class of all facets $(0, 1, 2)$ -valued coefficients characterized in [7]. A simple example of P_{FS} polytope with such a non-rank facet is as follows. The original system contains six inequalities in three variables. In addition to the rank inequalities defined by the five maximal IISs ($\{3456\}$, $\{2345\}$, $\{1346\}$, $\{1246\}$, $\{1245\}$) and to the trivial $(0, 1)$ -bounding inequalities, the single additional constraint $x_1 + x_2 + x_3 + 2x_4 + x_5 + x_6 \leq 5$ is required to provide the full description.

We have also constructed numerous examples of facets of P_{FS} having coefficients larger than 2. These examples come from full descriptions of small-to-medium size problems which we have analyzed using the software PORTA.

Acknowledgement The authors would like to thank G. M. Ziegler for helpful discussions regarding the material of Section 3.

References

- [1] C. C. AGGARWAL, R. K. AHUJA, J. HAO, AND J. B. ORLIN, *Diagnosing infeasibilities in network flow problems*, Mathematical Programming, 81 (1998), pp. 263–280.
- [2] E. AMALDI, *From finding maximum feasible subsystems of linear systems to feed-forward neural network design*, PhD thesis, Dep. of Mathematics, EPF-Lausanne, 1994.
- [3] E. AMALDI AND V. KANN, *The complexity and approximability of finding maximum feasible subsystems of linear relations*, Theoretical Comput. Sci., 147 (1995), pp. 181–210.
- [4] ———, *On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems*, Theoretical Comput. Sci., 209 (1998), pp. 237–260.
- [5] S. ARORA, L. BABAI, J. STERN, AND Z. SWEEDYK, *The hardness of approximate optima in lattices, codes, and systems of linear equations*, J. Comput. Syst. Sci., 54 (1997), pp. 317–331.
- [6] A. BACHEM AND M. GRÖTSCHEL, *New aspects of polyhedral theory*, in Optimization and Operations Research, A. Bachem, ed., Modern Applied Mathematics, North Holland, 1982, ch. I.2, pp. 51 – 106.
- [7] E. BALAS AND S. M. NG, *On the set covering polytope: All the facets with coefficients in $\{0,1,2\}$* , Mathematical Programming, 43 (1989), pp. 57–69.
- [8] L. BLUM, F. CUCKER, M. SHUB, AND S. SMALE, *Complexity and Real Computation*, Springer-Verlag, 1997.
- [9] J. BOKOWSKI AND B. STURMFELS, *Computational Synthetic Geometry*, no. 1355 in Lecture Notes in Mathematics, Springer-Verlag, 1989.
- [10] N. CHAKRAVARTI, *Some results concerning post-infeasibility analysis*, Eur. J. Oper. Res., 73 (1994), pp. 139–143.
- [11] J. CHINNECK, *Computer codes for the analysis of infeasible linear programs*, J. Oper. Res. Soc., 47 (1996), pp. 61–72.
- [12] ———, *An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem*, Annals of Mathematics and Artificial Intelligence, 17 (1996), pp. 127–144.
- [13] ———, *Feasibility and viability*, in Advances in Sensitivity Analysis and Parametric Programming, T. Gál and H. Greenberg, eds., Kluwer Academic Publishers, 1997.
- [14] J. CHINNECK AND E. DRAVNIKS, *Locating minimal infeasible constraint sets in linear programs*, ORSA Journal on Computing, 3 (1991), pp. 157–168.
- [15] M. DELL’AMICO, F. MAFFIOLI, AND S. MARTELLO, *Annotated Bibliographies in Combinatorial Optimization*, John Wiley, 1997.
- [16] K. FAN, *On systems of linear inequalities*, in Linear Inequalities and Related Systems, H. W. Kuhn and A. W. Tucker, eds., no. 38 in Annals of Mathematical Studies, Princeton University Press, NJ, 1956, pp. 99–156.
- [17] J. GLEESON AND J. RYAN, *Identifying minimally infeasible subsystems of inequalities*, ORSA Journal on Computing, 2 (1990), pp. 61–63.

- [18] H. J. GREENBERG, *Consistency, redundancy, and implied equalities in linear systems*, Annals of Mathematics and Artificial Intelligence, 17 (1996), pp. 37–83.
- [19] H. J. GREENBERG AND F. H. MURPHY, *Approaches to diagnosing infeasible linear programs*, ORSA Journal on Computing, 3 (1991), pp. 253–261.
- [20] J. HÅSTAD, *Some optimal inapproximability results*, in Proc. Twenty-ninth Ann. ACM Symp. Theory of Comp., ACM, 1997, pp. 1–10.
- [21] M. LAURENT, *A generalization of antiwebs to independence systems and their canonical facets*, Mathematical Programming, 45 (1989), pp. 97–108.
- [22] O. L. MANGASARIAN, *Misclassification minimization*, J. of Global Optimization, 5 (1994), pp. 309–323.
- [23] B. MISHRA, *Computational real algebraic geometry*, in Handbook of Discrete and Computational Geometry, J. Goodman and J. O’Rourke, eds., CRC Press, 1997, ch. 29.
- [24] N. E. MNĚV, *The universality theorems on the classification problem of configuration varieties and convex polytopes varieties*, in Topology and Geometry – Rohlin Seminar, O. Y. Viro, ed., no. 1346 in Lecture Notes in Mathematics, Springer-Verlag, 1988, pp. 527 – 543.
- [25] T. S. MOTZKIN, *Beiträge zur Theorie der Linearen Ungleichungen*, PhD thesis, Basel, 1933.
- [26] M. PARKER, *A set covering approach to infeasibility analysis of linear programming problems and related issues*, PhD thesis, Dep. of Mathematics, University of Colorado at Denver, 1995.
- [27] M. PARKER AND J. RYAN, *Finding the minimum weight IIS cover of an infeasible system of linear inequalities*, Annals of Mathematics and Artificial Intelligence, 17 (1996), pp. 107–126.
- [28] J. RICHTER-GEERT, *Realization Spaces of Polytopes*, no. 1643 in Lecture Notes in Mathematics, Springer-Verlag, 1996.
- [29] J. RYAN, *Transversals of IIS-hypergraphs*, Congressus Numerantium, 81 (1991), pp. 17–22.
- [30] ———, *IIS-hypergraphs*, SIAM J. Disc. Math., 9 (1996), pp. 643–653.
- [31] G. M. ZIEGLER, *Lectures on Polytopes*, Springer-Verlag, 1994.

Valid Inequalities for Problems with Additive Variable Upper Bounds[★]

Alper Atamtürk¹, George L. Nemhauser², and Martin W. P. Savelsbergh²

¹ Department of Industrial Engineering and Operations Research,
University of California, Berkeley, CA 94720-1777, USA
`atamturk@ieor.berkeley.edu`

² School of Industrial and Systems Engineering,
Georgia Institute of Technology, Atlanta, GA 30332-0205, USA
`{gnemhaus}{mwps}@isye.gatech.edu`

Abstract. We study the facial structure of a polyhedron associated with the single node relaxation of network flow problems with additive variable upper bounds. This type of structure arises, for example, in network design/expansion problems and in production planning problems with setup times. We first derive two classes of valid inequalities for this polyhedron and give the conditions under which they are facet-defining. Then we generalize our results through sequence independent lifting of valid inequalities for lower-dimensional projections. Our computational experience with large network expansion problems indicates that these inequalities are very effective in improving the quality of the linear programming relaxations.

1 Introduction

The single node fixed-charge flow polyhedron, studied by Padberg et al. [9] and Van Roy and Wolsey [12], arises as an important relaxation of many 0-1 mixed integer programming problems with fixed charges, including lot-sizing problems [4,10] and capacitated facility location problems [1]. The valid inequalities derived for the single node fixed-charge flow polyhedron have proven to be effective for solving these types of problems. Here we study a generalization of the single node fixed-charge flow polyhedron that arises as a relaxation of network flow problems with additive variable upper bounds, such as network design/expansion problems and production planning problems with setup times. We derive several classes of strong valid inequalities for this polyhedron. Our computational experience with network expansion problems indicates that these inequalities are very effective in improving the quality of the linear programming relaxations.

In a network design problem, given a network and demands on the nodes, we are interested in installing capacities on the edges of the network so that the total cost of flows and capacity installation is minimized. If some of the edges already

[★] This research is supported, in part, by NSF Grant DMI-9700285 to the Georgia Institute of Technology.

have positive capacities, the problem is called a network expansion problem. In many applications capacity is available in discrete quantities and has a cost structure that exhibits economies of scale [5,11]. The constraints of the mixed integer programming formulation of the network expansion problem for a single node are

$$\sum_{i \in M^+} y_i - \sum_{i \in M^-} y_i \leq b \quad (1)$$

$$y_i \leq u_i + \sum_{j \in N(i)} a_{ij} x_j, \quad i \in M. \quad (2)$$

Inequality (1) is the balance constraint of a node with inflow (M^+) and outflow (M^-) edges and demand b . The continuous variable y_i represents the flow on edge i , $i \in M = M^+ \cup M^-$. Inequalities (2) are the *additive variable upper bound* (AVUB) constraints on the flow variables. $N(i)$ is the index set of binary variables x_j representing the availability of resources that increase the capacity of edge i that has capacity u_i . For multi-commodity network expansion problems, it is possible to arrive at this single commodity relaxation by aggregating the balance constraints and the flow variables over commodities for a single node.

Additive variable upper bounds generalize the simple variable upper bounds in three respects. First, several binary variables additively increase the upper bound on the continuous variable. Second, the continuous variable is not necessarily restricted to zero when its additive variable bounds are zero. Third, we allow an overlap of additive variable upper bound variables, i.e. $N(i) \cap N(k) \neq \emptyset$ for $i, k \in M$. This situation typically occurs when capacities are installed on subsets of edges such as on cycles of the network (rings). Note that a simple variable upper bound constraint $y_i \leq u_i x_i$ is a special case of (2). We also point out that a variable lower bound constraint $l_i x_i \leq y_i$ can be put into a simple form of AVUB, $\bar{y}_i \leq (u_i - l_i) + l_i \bar{x}_i$, after complementing the binary variable x_i and the continuous variable y_i assuming that it has a finite upper bound u_i .

Multi-item production planning problems with setup times have the following constraints as part of their MIP formulations

$$\sum_{t' \leq t} y_{it'} \geq d_{it}, \quad \forall i, t \quad (3)$$

$$\sum_i y_{it} \leq u_t - \sum_i a_i x_{it}, \quad \forall t \quad (4)$$

where d_{it} denotes the demand for item i in period t , u_t the total production capacity in period t and a_i the setup time required for item i if the machine is setup for this item. Aggregating the demand constraints (3) and the production variables y_{it} for each period, we arrive at the same structure as in (1)-(2).

In the next section we introduce four classes of valid inequalities for

$$P = \{(x, y) \in \mathbb{B}^n \times \mathbb{R}_+^m : \text{subject to (1) and (2)}\}$$

and give conditions under which these inequalities are facet-defining for $\text{conv}(P)$. In Section 3 we present a summary of computational results on the use of the new inequalities in a branch-and-cut algorithm for network expansion problems.

2 Valid Inequalities

The proofs of the results given in the sequel are abbreviated or omitted due to space considerations. For detailed proofs and for further results and explanations, the reader is referred to Atamtürk [2].

Let $N = \{1, 2, \dots, n\}$ be the index set of binary variables and $N(S)$ be the subset of N appearing in the additive variable upper bound constraints associated with $S \subseteq M = \{1, 2, \dots, m\}$. For notational simplicity we use $N(i)$ for $N(\{i\})$. We define $u(S) = \sum_{i \in S} \left(u_i + \sum_{j \in N(i)} a_{ij} \right)$ for $S \subseteq M$ and $a_j(S) = \sum_{i \in S} a_{ij}$ for $j \in N(S)$. Again for notational simplicity we use $u(i)$ for $u(\{i\})$. Throughout we make the following assumptions on the data of the model:

- (A.1) $a_{ij} > 0$ for all $i \in M$, $j \in N(i)$.
- (A.2) $u(i) > 0$ for all $i \in M$.
- (A.3) $b + u(M^-) > 0$.
- (A.4) $u(i) - a_{ij} \geq 0$ for all $i \in M$, $j \in N(i)$.
- (A.5) $b + u(M^-) - a_j(M^-) \geq 0$ for all $j \in N$.

Assumptions (A.2-A.5) are made without loss of generality. If $u(i) < 0$ or $b + u(M^-) < 0$, then $P = \emptyset$. If $u(i) = 0$ ($b + u(M^-) = 0$), then $y_i = 0$ ($y_i = 0$, $i \in M^+$) in every feasible solution and can be eliminated. Similarly if $u(i) - a_{ij} < 0$ or $b + u(M^-) - a_j(M^-) < 0$, then $x_j = 1$ in every feasible solution and can be eliminated. Note that given (A.1), if $N(i) \neq \emptyset$ for all $i \in M$, then (A.4) implies (A.2) and (A.5) implies (A.3). Assumption (A.1) is made for convenience. Results presented in the sequel can easily be generalized to the case with $a_{ij} < 0$. Note that, for a particular $j \in N$ if $a_{ij} < 0$ for all $i \in M$, then x_j can be complemented to satisfy (A.1). If there is no overlap of additive variable upper bounds, i.e. $N(i) \cap N(k) = \emptyset$ for all $i, k \in M$, then $M(j)$ is singleton for all $j \in N$ and (A.1) can be satisfied by complementing the binary variables when $a_{ij} < 0$.

Proposition 1. *$\text{Conv}(P)$ is full-dimensional.*

2.1 Additive Flow Cover Inequalities

For $C^+ \subseteq M^+$ and $C^- \subseteq M^-$, (C^+, C^-) is said to be a *flow cover* if $\lambda = u(C^+) - b - u(C^-) > 0$. For a flow cover (C^+, C^-) , let $L^- \subseteq M^- \setminus C^-$ be such that $\gamma = \sum_{i \in L^-} u_i < \lambda$ and $K = M^- \setminus (C^- \cup L^-)$. Then, the *additive flow cover inequality* is

$$\sum_{i \in C^+} y_i + \sum_{j \in N(C^+)} (a_j(C^+) - \lambda + \gamma)^+ (1 - x_j) - \sum_{j \in N(L^-)} \min\{a_j(L^-), \lambda - \gamma\} x_j - \sum_{i \in K} y_i \leq b + u(C^-) + \gamma. \quad (5)$$

Proposition 2. *The additive flow cover inequality (5) is valid for P .*

Proof. Let $(\bar{x}, \bar{y}) \in P$ and $T = \{j \in N : \bar{x}_j = 0\}$. Also define $N(C^+)^+ = \{j \in N(C^+) : a_j(C^+) > \lambda - \gamma\}$, $N(L^-)^+ = \{j \in N(L^-) : a_j(L^-) > \lambda - \gamma\}$, and $N(L^-)^- = N(L^-) \setminus N(L^-)^+$. For $(\bar{x}, \bar{y})$ the left hand side of (5), *lhs*, equals

$$\sum_{i \in C^+} \bar{y}_i + \sum_{j \in N(C^+) \cap T} (a_j(C^+) - \lambda + \gamma)^+ - \sum_{j \in N(L^-) \setminus T} \min\{a_j(L^-), \lambda - \gamma\} - \sum_{i \in K} \bar{y}_i.$$

If $(N(C^+)^+ \cap T) \cup (N(L^-)^+ \setminus T) = \emptyset$, then

$$lhs \leq \sum_{i \in C^+} \bar{y}_i - \sum_{j \in N(L^-)^- \setminus T} a_j(L^-) - \sum_{i \in K} \bar{y}_i \leq b + u(C^-) + \gamma.$$

To see that the second inequality is valid, observe that $\sum_{i \in C^+} y_i - \sum_{i \in L^-} y_i - \sum_{i \in K} y_i \leq b + u(C^-)$ is valid for P and $\sum_{i \in L^-} y_i \leq u(L^-) - \sum_{j \in N(L^-)^+} a_j(L^-) - \sum_{j \in N(L^-) \cap T} a_j(L^-)$ is valid for $(\bar{x}, \bar{y})$ since $N(L^-)^+ \subseteq T$. Adding these two inequalities gives the result. Now, suppose $(N(C^+)^+ \cap T) \cup (N(L^-)^+ \setminus T) \neq \emptyset$. Then

$$\begin{aligned} lhs &\leq u(C^+) - \sum_{j \in N(C^+) \cap T} a_j(C^+) + \sum_{j \in N(C^+) \cap T} a_j(C^+) + \\ &\quad \sum_{j \in N(C^+) \cap T} (\gamma - \lambda) - \sum_{j \in N(L^-)^+ \setminus T} (\lambda - \gamma) - \sum_{j \in N(L^-)^- \setminus T} a_j(L^-) \\ &\leq u(C^+) - \lambda + \gamma - (\lambda - \gamma)[|N(C^+)^+ \cap T| + |N(L^-)^+ \setminus T| - 1] \\ &\leq b + u(C^-) + \gamma. \quad \square \end{aligned}$$

Remark 1. For the single node fixed-charge flow model, where (2) is replaced with $y_i \leq u_i x_i$, the additive flow cover inequality reduces to the flow cover inequality [12]

$$\sum_{i \in C^+} y_i + \sum_{i \in C^+} (u_i - \lambda)^+(1 - x_i) - \sum_{i \in L^-} \min\{u_i, \lambda\} x_i - \sum_{i \in K} y_i \leq b + \sum_{i \in C^-} u_i.$$

Proposition 3. *The additive flow cover inequality (5) is facet-defining for $\text{conv}(P)$ if the following five conditions are satisfied.*

1. $C^- = \emptyset$,
2. $\max_{j \in N(C^+)} a_j(C^+) > \lambda - \gamma$,
3. $a_j(L^-) > \lambda - \gamma$ for some $j \in N(i)$ for all $i \in L^-$ with $u_i = 0$,
4. $u_i \geq 0$ for all $i \in L^-$,
5. $N(L^-) \cap N(M \setminus L^-) = \emptyset$.

Note that if there is no overlap of additive variable upper bounds among continuous variables, then Condition 5 is trivially satisfied.

2.2 Additive Flow Packing Inequalities

Next we give the second class of valid inequalities for P . For $C^+ \subseteq M^+$ and $C^- \subseteq M^-$, (C^+, C^-) is said to be a *flow packing* if $\mu = -\lambda = b + u(C^-) - u(C^+) > 0$. For a flow packing (C^+, C^-) , let $L^+ \subseteq M^+ \setminus C^+$ be such that $\gamma = \sum_{i \in L^+} u_i < \mu$ and $K = M^- \setminus C^-$. Then the *additive flow packing inequality* is

$$\sum_{i \in C^+ \cup L^+} y_i - \sum_{j \in N(L^+)} \min\{a_j(L^+), \mu - \gamma\} x_j + \sum_{j \in N(C^-)} (a_j(C^-) - \mu + \gamma)^+(1 - x_j) - \sum_{i \in K} y_i \leq u(C^+) + \gamma. \quad (6)$$

Proposition 4. *The additive flow packing inequality (6) is valid for P .*

Proof. Let $(\bar{x}, \bar{y}) \in P$ and $T = \{j \in N : \bar{x}_j = 0\}$. Also let $N(C^-)^+ = \{j \in N(C^+) : a_j(C^-) > \mu - \gamma\}$, $N(L^+)^+ = \{j \in N(C^+) : a_j(L^+) > \mu - \gamma\}$, and $N(L^+)^- = N(L^+) \setminus N(L^+)^+$. For $(\bar{x}, \bar{y})$ the left hand side of (6), *lhs*, equals

$$\sum_{i \in C^+ \cup L^+} \bar{y}_i - \sum_{j \in N(L^+) \setminus T} \min\{a_j(L^+), \mu - \gamma\} + \sum_{j \in N(C^-)^+ \cap T} (a_j(C^-) - \mu + \gamma) - \sum_{i \in K} \bar{y}_i.$$

If $(N(L^+)^+ \setminus T) \cup (N(C^-)^+ \cap T) = \emptyset$, then

$$lhs = \sum_{i \in C^+ \cup L^+} \bar{y}_i - \sum_{j \in N(L^+)^- \setminus T} a_j(L^+) - \sum_{i \in K} \bar{y}_i \leq u(C^+) + \gamma.$$

Otherwise,

$$\begin{aligned} lhs &\leq b + u(C^-) - \sum_{j \in N(C^-) \cup T} a_j(C^-) - \sum_{j \in N(L^+)^+ \setminus T} (\mu - \gamma) \\ &\quad \sum_{j \in N(L^+)^- \setminus T} a_j(L^+) + \sum_{j \in N(C^-)^+ \cap T} (a_j(C^-) - \mu + \gamma) \\ &\leq b + u(C^-) - \mu + \gamma - (\mu - \gamma)[|N(L^+)^+ \setminus T| + |N(C^-)^+ \cap T| - 1] \\ &\leq u(C^+) + \gamma. \quad \square \end{aligned}$$

Remark 2. For the single node fixed-charge flow model, the additive flow packing inequality reduces to the flow packing inequality [3]

$$\sum_{i \in C^+ \cup L^+} y_i - \sum_{i \in L^+} \min\{u_i, \mu\} x_i + \sum_{i \in C^-} (u_i - \mu)^+(1 - x_i) - \sum_{i \in K} y_i \leq \sum_{i \in C^+} u_i.$$

Proposition 5. *The additive flow packing inequality (6) is facet-defining for $\text{conv}(P)$ if the following five conditions are satisfied.*

1. $C^+ = \emptyset$,
2. $u(M^-) + b > a_j(C^-) > \mu - \gamma$ for some $j \in N(C^-)$,
3. $a_j(L^+) > \mu - \gamma$ for some $j \in N(i)$ for all $i \in L^+$ with $u_i = 0$,
4. $u_i \geq 0$ for all $i \in L^+$,
5. $N(L^+) \cap N(M \setminus L^+) = N(C^-) \cap N(K) = \emptyset$.

2.3 Generalized Additive Flow Cover Inequalities

In order to derive more general classes of valid inequalities for P , we fix a subset of the binary variables to zero, derive a valid inequality for the resulting projection, and then lift this inequality with the variables that are fixed to zero. More precisely, let $F \subseteq N$ and consider the projection

$$P_F = \{(x, y) \in \mathbb{B}^{n-|F|} \times \mathbb{R}_+^m : \sum_{i \in M^+} y_i - \sum_{i \in M^-} y_i \leq b, \\ y_i \leq u_i + \sum_{j \in N(i) \setminus F} a_{ij} x_j, \quad i \in M\}$$

of P obtained by fixing $x_j = 0$ for all $j \in F$. We assume that $\text{conv}(P_F)$ is full-dimensional.

For $S \subseteq M$, let $\bar{u}(S) = \sum_{i \in S} (u_i + \sum_{j \in N(i) \setminus F} a_{ij})$. Let $C^+ \subseteq M^+$ and $C^- \subseteq M^-$ such that $\lambda = \bar{u}(C^+) - b - \bar{u}(C^-) > 0$ and $L^- \subseteq M^- \setminus C^-$ such that $\gamma = \sum_{i \in L^-} u_i < \lambda$. Then, from Section 2.1 we have the following valid additive flow cover inequality for P_F

$$\sum_{i \in C^+} y_i + \sum_{j \in N(C^+) \setminus F} (a_j(C^+) - \lambda + \gamma)^+(1 - x_j) - \\ \sum_{j \in N(L^-) \setminus F} \min\{a_j(L^-), \lambda - \gamma\} x_j - \sum_{i \in K} y_i \leq b + \bar{u}(C^-) + \gamma. \quad (7)$$

Note that inequality (7) is not necessarily valid for P . We assume that the conditions of Proposition 3 are satisfied and hence (7) is facet-defining for $\text{conv}(P_F)$. In order to derive a generalized additive flow cover inequality for P , we lift (7) in two phases. In the first phase we lift the inequality with the variables in $N(L^- \cup C^-) \cap F$. Then in the second phase we lift the resulting inequality with the variables in $N(C^+) \cap F$. When lifting the variables in phases, for convenience, we make the following assumption:

$$(A.6) \quad (N(C^+) \cap F) \cap (N(L^- \cup C^-) \cap F) = \emptyset.$$

Even if (A.6) is not satisfied, the lifted inequality is still valid for P , but it may not be facet-defining for $\text{conv}(P)$. Now, let (7) be lifted first with variable x_l , $l \in N(L^- \cup C^-) \cap F$. Then the lifting coefficient associated with x_l is equal to

$$f(a_l(L^- \cup C^-)) = b + \bar{u}(C^-) + \gamma - \max_{(x, y) \in P_F \setminus l, x_l = 1} \left\{ \sum_{i \in C^+} y_i + \right. \\ \left. \sum_{j \in N(C^+) \setminus F} (a_j(C^+) - \lambda + \gamma)^+(1 - x_j) - \sum_{j \in N(L^-) \setminus F} \min\{a_j(L^-), \lambda - \gamma\} x_j - \sum_{i \in K} y_i \right\}.$$

Since (7) satisfies the conditions of Proposition 3, it follows that $\bar{u}(C^+) > \lambda - \gamma$ or equivalently $b + \bar{u}(C^-) + \gamma > 0$. Then the lifting problem has an optimal

solution such that $y_i = 0$ for all $i \in (M^+ \setminus C^+) \cup K$. Let $(\bar{x}, \bar{y})$ be such an optimal solution and let $S = \{j \in N(C^+) \setminus F : \bar{x}_j = 0\}$ and $T = \{j \in N(L^-) \setminus F : \bar{x}_j = 1\}$. Clearly, we may assume that $S \subseteq \{j \in N(C^+) \setminus F : a_j(C^+) > \lambda - \gamma\}$ and $T \subseteq \{j \in N(L^-) \setminus F : a_j(L^-) > \lambda - \gamma\}$; otherwise we can obtain a solution with the same or better objective value by considering a subset of S or T satisfying these conditions. There are two cases to consider when determining the value of $f(a_l(L^- \cup C^-))$ depending on how $\sum_{i \in C^+} y_i$ is bounded in an optimal solution. We analyze $f(a_l(L^- \cup C^-))$ separately for each case.

Case 1: $\lambda - \gamma \leq \sum_{j \in S} a_j(C^+) + \sum_{j \in T} a_j(L^-) + a_l(L^- \cup C^-)$.

$$\begin{aligned} f(a_l(L^- \cup C^-)) &= b + \bar{u}(C^-) + \gamma - [\bar{u}(C^+) - \sum_{j \in S} a_j(C^+) + \\ &\quad \sum_{j \in S} (a_j(C^+) - \lambda + \gamma) - \sum_{j \in T} (\lambda - \gamma)] \\ &= (|S \cup T| - 1)(\lambda - \gamma). \end{aligned}$$

Case 2: $\lambda - \gamma > \sum_{j \in S} a_j(C^+) + \sum_{j \in T} a_j(L^-) + a_l(L^- \cup C^-)$.

$$\begin{aligned} f(a_l(L^- \cup C^-)) &= b + \bar{u}(C^-) + \gamma - [b + \bar{u}(C^-) + \gamma + \sum_{j \in T} a_j(L^-) + \\ &\quad a_l(L^- \cup C^-) + \sum_{j \in S} (a_j(C^+) - \lambda + \gamma) - \sum_{j \in T} (\lambda - \gamma)] \\ &= |S \cup T|(\lambda - \gamma) - \sum_{j \in S} a_j(C^+) - \sum_{j \in T} a_j(L^-) - a_l(L^- \cup C^-). \end{aligned}$$

Observe that in Case 2, i.e. if the balance constraint is tight, $S = T = \emptyset$ since $a_l(L^- \cup C^-) \geq 0$ and by assumption $a_j(C^+) > \lambda - \gamma$ for all $j \in S$ and $a_j(L^-) > \lambda - \gamma$ for all $j \in T$. Also in Case 1, $f(a_l(L^- \cup C^-))$ is minimized when $S = T = \emptyset$. Then, we conclude that $f(a_l(L^- \cup C^-)) = -\min\{a_l(L^- \cup C^-), \lambda - \gamma\}$. It is easy to see that f is superadditive on $\mathbb{R}_-$, which implies that the lifting is sequence independent, that is the lifting function f remains unchanged as the projected variables in $N(L^- \cup C^-) \cap F$ are introduced to inequality (7) sequentially [7,13]. Therefore,

$$\begin{aligned} \sum_{i \in C^+} y_i + \sum_{j \in N(C^+) \setminus F} (a_j(C^+) - \lambda + \gamma)^+(1 - x_j) - \\ \sum_{j \in N(L^- \cup C^-) \cap F} \min\{a_j(L^- \cup C^-), \lambda - \gamma\} x_j - \\ \sum_{j \in N(L^-) \setminus F} \min\{a_j(L^-), \lambda - \gamma\} x_j - \sum_{i \in K} y_i \leq b + \bar{u}(C^-) + \gamma \end{aligned} \tag{8}$$

is a valid inequality for $P_{N(C^+) \cap F}$.

In the second phase, we lift inequality (8) with $x_l, l \in N(C^+) \cap F$. The lifting coefficient of x_l equals

$$g(a_l(C^+)) = b + \bar{u}(C^-) + \gamma - \max_{(x,y) \in P_{N(C^+) \cap F \setminus l, x_l=1}} \left\{ \sum_{i \in C^+} y_i + \sum_{j \in N(C^+) \setminus F} (a_j(C^+) - \lambda + \gamma)^+(1 - x_j) - \sum_{j \in N(L^-) \setminus F} \min\{a_j(L^-), \lambda - \gamma\} x_j - \sum_{j \in N(L^- \cup C^-) \cap F} \min\{a_j(L^- \cup C^-), \lambda - \gamma\} x_j - \sum_{i \in K} y_i \right\}.$$

The lifting problem has an optimal solution such that $y_i = 0$ for all $i \in (M^+ \setminus C^+) \cup K$, $x_j = 1$ for all $j \in N(C^+) \setminus F$ such that $a_j(C^+) \leq \lambda - \gamma$, $x_j = 0$ for all $j \in N(L^- \cup C^-) \cap F$ such that $a_j(L^- \cup C^-) \leq \lambda - \gamma$, and $x_j = 0$ for all $j \in N(L^-) \setminus F$ such that $a_j(L^-) \leq \lambda - \gamma$. Let $(\bar{x}, \bar{y})$ be such an optimal solution. Let $R = \{j \in N(L^- \cup C^-) \cap F : \bar{x}_j = 1\}$, $S = \{j \in N(C^+) \setminus F : \bar{x}_j = 0\}$, and $T = \{j \in N(L^-) \setminus F : \bar{x}_j = 1\}$. Again, there are two cases when determining the value of $g(a_l(C^+))$ depending on how $\sum_{i \in C^+} y_i$ is bounded in an optimal solution.

$$\text{Case 1: } \lambda - \gamma \leq \sum_{j \in S} a_j(C^+) + \sum_{j \in R} a_j(L^- \cup C^-) + \sum_{j \in T} a_j(L^-) - a_l(C^+).$$

$$\begin{aligned} g(a_l(C^+)) &= b + \bar{u}(C^-) + \gamma - [\bar{u}(C^+) - \\ &\quad \sum_{j \in S} a_j(C^+) + a_l(C^+) + \sum_{j \in S} (a_j(C^+) - \lambda + \gamma) - \sum_{i \in R \cup T} (\lambda - \gamma)] \\ &= (|R \cup S \cup T| - 1)(\lambda - \gamma) - a_l(C^+). \end{aligned}$$

$$\text{Case 2: } \lambda - \gamma > \sum_{j \in S} a_j(C^+) + \sum_{j \in R} a_j(L^- \cup C^-) + \sum_{j \in T} a_j(L^-) - a_l(C^+).$$

$$\begin{aligned} g(a_l(C^+)) &= b + \bar{u}(C^-) + \gamma - [b + \bar{u}(C^-) + \gamma + \sum_{j \in R} a_j(L^- \cup C^-) + \\ &\quad \sum_{j \in T} a_j(L^-) + \sum_{j \in S} (a_j(C^+) - \lambda + \gamma) - \sum_{j \in R \cup T} (\lambda - \gamma)] \\ &= |R \cup S \cup T|(\lambda - \gamma) - \sum_{j \in S} a_j(C^+) - \sum_{j \in R} a_j(L^- \cup C^-) - \sum_{j \in T} a_j(L^-). \end{aligned}$$

Now, let

$$v_j = \begin{cases} a_j(C^+), & \text{if } j \in N(C^+) \setminus F, \\ a_j(L^- \cup C^-), & \text{if } j \in N(L^- \cup C^-) \cap F, \\ a_j(L^-), & \text{if } j \in N(L^-) \setminus F, \end{cases}$$

$$x'_j = \begin{cases} 1 - x_j, & \text{if } j \in N(C^+) \setminus F, \\ x_j, & \text{if } j \in N(L^-) \cup (N(C^-) \cap F), \end{cases}$$

and $\{j_1, j_2, \dots, j_r\} = \{j \in (N(C^+) \setminus F) \cup N(L^-) \cup (N(C^-) \cap F) : v_j > \lambda - \gamma\}$ such that $v_{j_k} \geq v_{j_{k+1}}$ for $k = 1, 2, \dots, r-1$. We also define the partial sums $w_0 = 0$, $w_k = \sum_{i=1}^k v_{j_i}$ for $k = 1, 2, \dots, r$.

It is not hard to show that there is a monotone optimal solution to the lifting problem. That is, there exists an optimal solution such that $\bar{x}'_{j_k} \geq \bar{x}'_{j_{k+1}}$ for $k = 1, 2, \dots, r-1$. Therefore $g(a_l(C^+))$ can be expressed in a closed form as follows:

$$g(a_l(C^+)) = \begin{cases} k(\lambda - \gamma) - a_l(C^+), & w_k < a_l(C^+) \leq w_{k+1} - \lambda + \gamma, \\ & k = 0, 1, \dots, r-1, \\ k(\lambda - \gamma) - w_k, & w_k - \lambda + \gamma < a_l(C^+) \leq w_k, \quad k = 1, 2, \dots, r, \\ r(\lambda - \gamma) - w_r, & w_r < a_l(C^+). \end{cases}$$

It can be shown that g is superadditive on $\mathbb{R}_-$, which implies that the lifting function g remains unchanged as the projected variables in $N(C^+) \cap F$ are introduced to inequality (8) sequentially [7,13]. Hence we have the following result.

Proposition 6. *The generalized additive flow cover inequality*

$$\begin{aligned} \sum_{i \in C^+} y_i + \sum_{j \in N(C^+) \setminus F} (a_j(C^+) - \lambda + \gamma)^+(1 - x_j) + \\ \sum_{j \in N(C^+) \cap F} \alpha_j x_j - \sum_{j \in N(L^- \cup C^-) \cap F} \min\{a_j(L^- \cup C^-), \lambda - \gamma\} x_j - \quad (9) \\ \sum_{j \in N(L^-) \setminus F} \min\{a_j(L^-), \lambda - \gamma\} x_j - \sum_{i \in K} y_i \leq b + \bar{u}(C^-) + \gamma \end{aligned}$$

with

$$\alpha_j = \begin{cases} k(\lambda - \gamma) - a_j(C^+), & w_k < a_j(C^+) \leq w_{k+1} - \lambda + \gamma, \quad k = 0, 1, \dots, r-1, \\ k(\lambda - \gamma) - w_k, & w_k - \lambda + \gamma < a_j(C^+) \leq w_k, \quad k = 1, 2, \dots, r, \\ r(\lambda - \gamma) - w_r, & w_r < a_j(C^+). \end{cases}$$

is valid for P .

Proposition 7. *The generalized additive flow cover inequality (9) is facet-defining for $\text{conv}(P)$ if (7) is facet-defining for $\text{conv}(P_F)$.*

2.4 Generalized Additive Flow Packing Inequalities

Here we generalize the additive flow packing inequalities with the same approach taken in Section 2.3 for the additive flow cover inequalities. Consider the projection P_F of P introduced in Section 2.3. Let $C^+ \subseteq M^+$ and $C^- \subseteq M^-$ such that $\mu = b + \bar{u}(C^-) - \bar{u}(C^+) > 0$ and $L^+ \subseteq M^+ \setminus C^+$ such that $\gamma = \sum_{i \in L^+} u_i < \mu$.

Then from Section 2.2 we have the following valid additive flow packing inequality for P_F

$$\begin{aligned} \sum_{i \in C^+ \cup L^+} y_i - \sum_{j \in N(L^+) \setminus F} \min\{a_j(L^+), \mu - \gamma\} x_j + \\ \sum_{j \in N(C^-) \setminus F} (a_j(C^-) - \mu + \gamma)^+(1 - x_j) - \sum_{i \in K} y_i \leq \bar{u}(C^+) + \gamma. \end{aligned} \quad (10)$$

We assume that the conditions of Proposition 5 are satisfied and hence (10) is facet-defining for $\text{conv}(P_F)$. To introduce the variables in F into inequality (10), we lift (10) in two phases. First we lift the inequality with the variables in $N(C^+ \cup L^+) \cap F$. Then in the second phase we lift the resulting inequality with variables in $N(C^-) \cap F$. When employing this two phase lifting procedure, for convenience, we assume that

$$(A.7) \quad (N(L^+ \cup C^+) \cap F) \cap (N(C^-) \cap F) = \emptyset.$$

The lifting of inequality (10) proceeds similar to the lifting of inequality (7). Therefore, we only give the final result here.

Proposition 8. *The generalized additive flow packing inequality*

$$\begin{aligned} \sum_{i \in C^+ \cup L^+} y_i - \sum_{j \in N(L^+) \setminus F} \min\{a_j(L^+), \mu - \gamma\} x_j - \\ \sum_{j \in N(C^+ \cup L^+) \cap F} \min\{a_j(C^+ \cup L^+), \mu - \gamma\} x_j + \sum_{j \in N(C^-) \cap F} \alpha_j x_j - \\ \sum_{j \in N(C^-) \setminus F} (a_j - \mu + \gamma)^+(1 - x_j) - \sum_{i \in K} y_i \leq \bar{u}(C^+) + \gamma \end{aligned} \quad (11)$$

with

$$\alpha_j = \begin{cases} k(\mu - \gamma) - a_j(C^-), & w_k < a_j(C^-) \leq w_{k+1} - \mu + \gamma, \quad k = 0, 1, \dots, r-1, \\ k(\mu - \gamma) - w_k, & w_k - \mu + \gamma < a_j(C^-) \leq w_k, \quad k = 1, 2, \dots, r, \\ r(\mu - \gamma) - w_r, & w_r < a_j(C^-), \end{cases}$$

is valid for P .

Proposition 9. *The generalized additive flow packing inequality (11) is facet-defining for $\text{conv}(P)$ if (10) is facet-defining for $\text{conv}(P_F)$.*

3 Computational Results

In this section, we present our computational results on solving network expansion problems with a branch-and-cut algorithm. We implemented heuristic separation algorithms for the generalized additive flow cover and flow packing inequalities for the single node relaxation of the problem. We also used the lifted

cover inequalities [6] for surrogate 0-1 knapsack relaxations of the single node relaxation, where the continuous flow variables are replaced with either their 0-1 additive variable upper bound variables or with their lower bounds. The branch-and-cut algorithm was implemented with MINTO [8] (version 3.0) using CPLEX as LP solver (version 6.0). All of the experiments were performed on a SUN Ultra 10 workstation with a one hour CPU time limit and a 100,000 nodes search tree size limit.

We present a summary of two experiments. The first experiment is performed to test the effectiveness of the cuts in solving a set of randomly generated network expansion problems with 20 vertices and 70% edge density. The instances were solved using MINTO first with its default settings and then with the above mentioned cutting planes generated throughout the search tree. In Table 1, we report the number of AVUB variables per flow variable (**avubs**) and the average values for the LP relaxation at the root node of the search tree (**zroot**), the best lower bound (**zlb**) and the best upper bound (**zub**) on the optimal value at termination, the percentage gap between **zlb** and **zub** (**endgap**), the number of generalized additive flow cover cuts (**gafcov**), generalized additive flow packing cuts (**gafpack**), surrogate knapsack cover cuts (**skcov**) added, the number of nodes evaluated (**nodes**), and the CPU time elapsed in seconds (**time**) for five random instances. While none of the problems could be solved to optimality without adding the cuts within 100,000 nodes, all of the problems were solved easily when the cuts were added. We note that MINTO does not generate any flow cover inequalities for these problem, since it does not recognize that additive variable upper bounds can be relaxed to simple variable upper bounds. Observe that the addition of the cuts improves the lower bounds as well as the upper bounds significantly, which leads to much smaller search trees and overall solution times. Table 1 clearly shows the effectiveness of the cuts.

Table 1. Effectiveness of cuts: 20 vertices.

	avubs	zroot	zlb	zub	endgap	gafcov	gafpack	skcov	nodes	time
without cuts	2	9.49	10.22	16.80	39.00	0	0	0	100,000	1386
	4	12.74	16.99	25.60	33.86	0	0	0	100,000	1018
	8	2.77	11.51	58.40	79.15	0	0	0	100,000	1859
with cuts	2	15.30	15.40	15.40	0.00	46	42	15	6	1
	4	23.73	25.20	25.20	0.00	55	33	47	73	5
	8	36.36	38.40	38.40	0.00	47	34	41	121	9

In the next experiment, we solved larger instances of the network expansion problem with 20% edge density to find out the sizes of instances that can be solved with the branch-and-cut algorithm. The results of this experiment are summarized in Table 2, where we present the number of AVUB variables per flow variable (**avubs**), the average values for the percentage difference between the initial LP relaxation and **zub** (**initgap**), the percentage difference between the LP relaxation after the cuts are added at the root node and **zub** (**rootgap**),

in addition to **endgap**, **gafcov**, **gafpack**, **skcov**, **nodes**, and **time** for five random instances with 50, 100 and 150 vertices. We note that these problems are much larger than ones for which computations are provided in the literature [5,11]. Although all of the instances with 50 vertices could be solved to optimality, for the larger instances the gap between the best lower bound and the best upper bound could not be completely closed for most of the problems with 4 or 8 **avubs** within an hour of CPU time. Nevertheless, the improvement in LP relaxations is significant, ranging between 50% and 98%.

Table 2. Performance of the branch-and-cut algorithm.

vertices	avubs	initgap	rootgap	endgap	gafcov	gafpack	skcov	nodes	time
50	1	25.09	1.05	0.00	48	56	16	24	2
	2	20.12	2.20	0.00	115	102	38	110	11
	4	36.34	3.31	0.00	192	139	106	721	49
	8	92.06	3.72	0.00	416	216	164	7712	960
100	1	14.80	0.94	0.00	182	148	41	44	39
	2	12.27	4.44	0.00	590	324	191	1236	961
	4	38.04	3.74	2.37	707	382	448	2437	2171
	8	92.75	9.15	8.75	828	408	509	1717	3600
150	1	16.56	0.29	0.00	480	307	57	346	586
	2	10.81	5.39	4.34	586	395	318	813	2965
	4	43.69	12.22	12.22	862	527	672	711	3600
	8	93.08	15.02	15.02	659	449	503	477	3600

For most of the unsolved problems, the best lower bound and the best upper bound were found at the root node in a few minutes; no improvement in the gap was observed later in the search tree. For instance **nexp.100.8.5** the value of the initial LP relaxation was 18.23. After adding 916 cuts in 42 rounds the root LP relaxation improved to 220.81, which was in fact the best lower bound found in the search tree, in 617 seconds. The best upper bound 259 was again found at the root node by a simple heuristic which installs the least cost integral capacity feasible for the flow on each edge provided by the LP relaxation. Therefore, for the unsolved problems it is likely that the actual duality gaps of the improved LP relaxations are much smaller.

More detailed experiments to compare the relative effectiveness of the different classes of cuts revealed that the generalized additive flow cover inequalities were the most effective, and that the lifted surrogate knapsack inequalities were more effective than the generalized additive flow packing inequalities. However, the use of all three classes of cuts delivered the best performance in most cases. From these computational results, we conclude that the valid inequalities derived from the single node relaxations are very effective in improving the LP bounds for network design/expansion problems.

References

1. K. Aardal, Y. Pochet, and L. A. Wolsey. Capacitated facility location: Valid inequalities and facets. *Mathematics of Operations Research*, 20:562–582, 1995.
2. A. Atamtürk. *Conflict graphs and flow models for mixed-integer linear optimization problems*. PhD thesis, ISyE, Georgia Institute of Technology, Atlanta, USA, 1998.
3. A. Atamtürk. Flow packing facets of the single node fixed-charge flow polytope. Technical report, IEOR, University of California at Berkeley, 1998.
4. I. Barany, T. J. Van Roy, and L. A. Wolsey. Uncapacitated lot sizing: The convex hull of solutions. *Mathematical Programming Study*, 22:32–43, 1984.
5. D. Bienstock and O. Günlük. Capacitated network design - Polyhedral structure and computation. *INFORMS Journal on Computing*, 8:243–259, 1996.
6. Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Lifted knapsack covers inequalities for 0-1 integer programs: Computation. Technical Report LEC-94-9, Georgia Institute of Technology, Atlanta GA, 1994. (to appear in *INFORMS Journal on Computing*).
7. Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Sequence independent lifting. Technical Report LEC-95-08, Georgia Institute of Technology, Atlanta, 1995.
8. G. L. Nemhauser, M. W. P. Savelsbergh, and G. S. Sigismondi. MINTO, a Mixed INTeger Optimizer. *Operations Research Letters*, 15:47–58, 1994.
9. M. W. Padberg, T. J. Van Roy, and L. A. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 32:842–861, 1984.
10. Y. Pochet. Valid inequalities and separation for capacitated economic lot sizing. *Operations Research Letters*, 7:109–115, 1988.
11. M. Stoer and G. Dahl. A polyhedral approach to multicommodity survivable network. *Numerische Mathematik*, 68:149–167, 1994.
12. T. J. Van Roy and L. A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14:199–213, 1986.
13. L. A. Wolsey. Valid inequalities and superadditivity for 0/1 integer programs. *Mathematics of Operations Research*, 2:66–77, 1977.

A Min-Max Theorem on Feedback Vertex Sets (Preliminary Version)

Mao-cheng Cai^{1,*,**}, Xiaotie Deng^{2,**} and Wenan Zang^{3,***}

¹ Institute of Systems Science, Academia Sinica
Beijing 100080, P. R. China
`caimc@bamboo.iss.ac.cn`

² Department of Computer Science, City University of Hong Kong
Hong Kong, P. R. China
`deng@cs.cityu.edu.hk`

³ Department of Mathematics, The University of Hong Kong
Hong Kong, P. R. China
`WZANG@maths.hku.hk`

Abstract. We establish a necessary and sufficient condition for the linear system $\{x : Hx \geq e, x \geq 0\}$ associated with a bipartite tournament to be *TDI*, where H is the cycle-vertex incidence matrix and e is the all-one vector. The consequence is a min-max relation on packing and covering cycles, together with strongly polynomial time algorithms for the feedback vertex set problem and the cycle packing problem on the corresponding bipartite tournaments. In addition, we show that the feedback vertex set problem on general bipartite tournaments is NP-complete and approximable within 3.5 based on the max-min theorem.

Key words. feedback vertex set, bipartite tournament, totally dual integrality, min-max relation, approximation algorithm.

AMS subject classification. 68Q25, 68R10.

1 Introduction

The basic theme of polyhedral combinatorics is the application of linear programming duality theory to combinatorial problems, the end product of which is often a combinatorial min-max result. In addition to its esthetical value and theoretical interest, a combinatorial min-max relation usually leads to algorithmic solvability of the corresponding optimization problem. The model of the totally dual integral (abbreviated *TDI*) systems proposed by Edmonds and Giles [2,3] serves as a general framework for establishing min-max relations for

* Research partially supported by the National Natural Science Foundation of China.

** Research supported in part by a RGC CERG grant and a SRG grant of City University of Hong Kong

*** Supported in part by RGC grant 338/024/0009.

various combinatorial problems, such as Fulkerson's optimum arborescence theorem, Lucchesi-Younger theorem, Edmonds' matroid intersection theorem, and Edmonds' matching polyhedron theorem [7]; it is also closely related to many other models for the same purpose [6].

A rational linear system $\{x : Ax \geq b, x \geq 0\}$ is called *totally dual integral*, if the maximum $\text{Max } \{y^T b \mid y^T A \leq c^T, y \geq 0\}$ has an integral optimum solution y for every integral vector c for which the maximum is finite. As shown by Edmonds and Giles [2], if $\{Ax \geq b, x \geq 0\}$ is *TDI*, b is integral, and $\text{Min } \{c^T x \mid Ax \geq b, x \geq 0\}$ is finite, then it has an integral optimal solution x . Giles and Pulleyblank showed that any rational polyhedron P has a *TDI* system $P = \{x : Ax \leq b\}$ representation with A integral. Moreover, b can be chosen to be integral if and only if P is integral [4]. Usually, a polyhedron can be defined by different systems of linear inequalities. It was proved by Schrijver [8] that if a rational polyhedron P is of full dimension, then P has a unique minimal *TDI* system representation $P = \{x : Ax \leq b\}$ with A and b integral if and only if P is integral.

Many combinatorial problems involve naturally defined classes of polyhedra, for instance, the vertex cover problem of graphs. Let A be the edge-vertex incidence matrix of a graph G . Then $\{x : Ax \geq e, x \in \{0, 1\}\}$ is the collection of vertex cover sets. A classical theorem asserts that the linear relaxation $\{x : Ax \geq e, x \geq 0\}$ is *TDI* if and only if G is bipartite. In a recent work [1], we investigated the feedback vertex set problem on tournaments using a similar approach. Given a digraph $D = (V, E)$ with weight on each vertex, a subset $S \subset V$ is called a *feedback vertex set* if $V(C) \cap S \neq \emptyset$ for any (directed) cycle C in D . The problem of finding a feedback vertex set with the minimum total weight is called the *feedback vertex set problem*. Let H be the triangle-vertex incidence matrix of a tournament, we [1] established a necessary and sufficient condition for $\{x : Hx \geq e, x \geq 0\}$ to be *TDI*. This allowed us to obtain a 2.5-approximation algorithm (based on the subgraph removal technique) for the minimum feedback vertex set problem on tournaments, improving a previous known algorithm with performance guarantee of three by Speckenmeyer [9].

We are interested in extending this approach to other problems. In this work, we study the feedback vertex set problem on bipartite tournaments, where a *bipartite tournament* is an orientation of a complete bipartite graph. We prove that a linear system associated with the minimum feedback vertex set problem on a bipartite tournament T is *TDI* if and only if T contains no F_1 nor F_2 (see Figure 1). We also give strongly polynomial time algorithms for the feedback vertex set problem and the cycle packing problem on these bipartite tournaments. In comparison with the previous work [1] for tournaments, this work requires deeper insight into the mathematical structure of bipartite tournaments. In fact, it would be a formidable task (if not impossible) to adapt the proof in the previous work to the more complicated problem studied here, and the present proof is much more concise and mathematically easier to understand.

In Section 2, we introduce notations and give a structural description of bipartite tournaments with no F_1 nor F_2 . In section 3, we proceed to investigate the maximum cycle-packing problem on these bipartite tournaments and prove

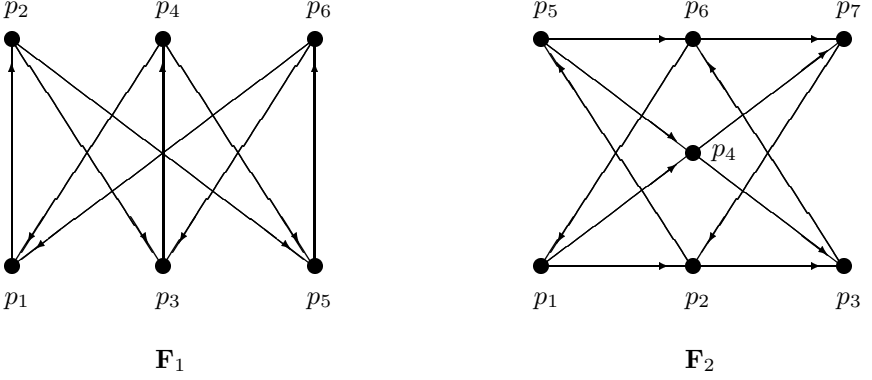


Figure 1: *Two Forbidden Bipartite Subtournaments*

that the packing problem has an integral optimum solution for any nonnegative integral weight function w on vertices. In Section 4, we establish a *TDI* system associated with the cycle-covering problem, which yields an integral optimum solution to the LP relaxation for the minimum cycle-covering problem on these bipartite tournaments. As a result, we obtain a min-max theorem—the cycle-packing number equals the cycle-covering number for any bipartite tournament with no F_1 nor F_2 . In addition, we present strongly polynomial time algorithms for the cycle-packing and cycle-covering problems. In section 5, we show the NP-completeness of the feedback set problem on general bipartite tournaments. Thus it is natural to consider the approximation problem. Clearly, a 4-approximation algorithm for this problem can be obtained by the primal-dual method [5]. Based on the *TDI* system, we shall be able to improve the ratio from 4 to 3.5. This exhibits another example of applying the duality structures of linear programs to approximation algorithms. In section 6, we conclude this paper with remarks and discussion.

2 Preliminaries

We consider a bipartite tournament $T = (V, E; w)$, with a weight function on the vertices $w : V \rightarrow R_+ = \{x : x \geq 0\}$. Let (u, v) denote the arc of T from vertex u to vertex v , and let $N^-(v) = \{u \in V \mid (u, v) \in E\}$ and $N^+(v) = \{u \in V \mid (v, u) \in E\}$, let $d(u, v)$ stand for the distance from u to v . Without loss of generality, let us assume henceforward that T under consideration is strongly connected for otherwise we may consider its components separately. Clearly, in bipartite tournaments, $S \subseteq V$ is a feedback vertex set if and only if S intersects every (directed) cycle of length four, denoted by C_4 . Thus the feedback vertex set problem is actually the C_4 -covering problem. Similarly, the cycle packing problem is actually the C_4 -packing problem. We shall focus on C_4 's in place of cycles in the remainder of the paper.

More formally, a C_4 -packing in T is a family of C_4 's (repetition is allowed) in T such that each vertex is contained in at most $w(v)$ of C_4 's in this family. A

maximum C_4 -packing in T is a C_4 -packing in T with the largest size. The C_4 -*packing number* of T is the size of a maximum C_4 -packing in T . A C_4 -*covering* in T is a vertex set $S \subseteq V$ that intersects each C_4 in T . The *size* of S , denoted by $w(S)$, is $\sum_{v \in S} w(v)$. A *minimum* C_4 -covering in T is a C_4 -covering with the smallest size; the C_4 -*covering number* of T is the size of a minimum C_4 -covering in T .

The C_4 -covering number of T is always greater than or equal to the C_4 -packing number of T . The situation in which the packing and covering numbers are equal is particularly interesting. We point out that equality does not necessarily hold on general bipartite tournaments: both F_1 and F_2 have C_4 -packing number of 1 and C_4 -covering number of 2. We shall demonstrate, based on the above-mentioned *TDI* system, that actually F_1 and F_2 are the only obstructions for our problem: if a bipartite tournament $T = (V, E)$ contains no F_1 nor F_2 , then the C_4 -packing number of T equals the C_4 -covering number of T .

Let us now present a structural description of the bipartite-tournaments with no F_1 nor F_2 , which will be used repeatedly later.

Lemma 2.1 *Let $T = (V, E)$ be a strongly connected bipartite tournament with no subdigraph isomorphic to F_1 nor F_2 . Then the vertex set V can be partitioned into $V_1, V_2, \dots, V_k$ for some k with $4 \leq k \leq |V|$, which have the following properties:*

- (a) *For $i = 1, 2, \dots, k$, V_i is independent;*
- (b) *For $i = 1, 2, \dots, k$, V_i admits a linear order $\prec$ such that for any arc $(u, v) \in E$, we have*
 - $(u, x) \in E$ whenever $v \prec x$ and
 - $(x, v) \in E$ whenever $x \prec u$.
- (c) *There is no arc between V_i and V_{i+2j} for $1 \leq i \leq k-2$ and $1 \leq j \leq \lfloor \frac{k-i}{2} \rfloor$; each arc between V_i and V_{i+1+2j} is directed from V_i to V_{i+1+2j} for $1 \leq i \leq k-3$ and $1 \leq j \leq \lfloor \frac{k-1-i}{2} \rfloor$.*

Proof. Let us reserve the symbol w for a vertex in T with *minimum indegree* throughout the proof. For $i = 0, 1, 2, \dots$, define

$$V_{i+1} := \{v \in V \mid d(v, w) = i\}.$$

Let k stand for the largest subscript with $V_k \neq \emptyset$. Then $V_1, V_2, \dots, V_k$ form a partition of V as T is strongly connected. Note that each V_i is independent for T is bipartite. In addition, $k \geq 4$ for $N^+(w) \neq \emptyset$.

We shall show that $V_1, V_2, \dots, V_k$ have the desired properties. For this purpose, observe that

(2.1) For each $v \in V_i$ with $2 \leq i \leq k$, $N^+(v) \cap V_{i-1} \neq \emptyset$ by the definition of V_i .

(2.2) For each $v \in V_i$ with $i = 2, 3$, $N^-(v) \cap V_{i+1} \neq \emptyset$, since otherwise, if $v \in V_2$, then v would be a source of D , contradicting the strong connectivity of

D ; if $v \in V_3$, then $d^-(v) < d^-(w)$ for T is bipartite, contradicting the choice of w .

(2.3) There is no arc between V_i and V_{i+2j} for $1 \leq i \leq k$ and $1 \leq j \leq \lfloor \frac{k-i}{2} \rfloor$ as T is bipartite.

(2.4) Each arc between V_i and V_{i+1+2j} is directed from V_i to V_{i+1+2j} for $1 \leq i \leq k-3$ by the definition of V_{i+1+2j} , $j \geq 1$.

Property (c) thus follows. To prove that $V_1, V_2, \dots, V_k$ enjoy properties (a) and (b), we make the following observations.

(2.5) There is no C_4 in $[V_i \cup V_{i+1}]$ for $i = 1, 2, \dots, k-1$.

Assume the contrary: some $[V_i \cup V_{i+1}]$ contains a C_4 . Let i be such smallest subscript and let $v_1 v_2 v_3 v_4 v_1$ be a C_4 with $v_1, v_3 \in V_i$ and $v_2, v_4 \in V_{i+1}$. Then $i \geq 2$. Now let us distinguish between two cases.

Case 1. $i = 2$. If there exists $v_5 \in V_4$ such that $(v_5, v_2), (v_5, v_4) \in E$, then $\{v_1, \dots, v_5, w\}$ induces an F_1 in T with vertex correspondence $v_1 \leftrightarrow p_2, v_2 \leftrightarrow p_3, v_3 \leftrightarrow p_4, v_4 \leftrightarrow p_1, v_5 \leftrightarrow p_6, w \leftrightarrow p_5$, a contradiction. Thus, by (2.2), there exist two vertices $v_5, v_6 \in V_4$ such that $(v_5, v_2), (v_6, v_4), (v_2, v_6), (v_4, v_5) \in E$. So $\{v_1, \dots, v_6, w\}$ induces an F_2 in T with correspondence $v_1 \leftrightarrow p_1, v_2 \leftrightarrow p_2, v_3 \leftrightarrow p_5, v_4 \leftrightarrow p_6, v_5 \leftrightarrow p_7, v_6 \leftrightarrow p_3, w \leftrightarrow p_4$, again a contradiction.

Case 2. $i \geq 3$. First observe that there exists $v \in V_{i-1}$ such that $(v_1, v), (v_3, v) \in E$. (Indeed, by (2.1) there exist $v, v' \in V_{i-1}$ such that $(v_1, v), (v_3, v') \in E$. Thus either $(v_1, v') \in E$ or $(v_3, v) \in E$ for otherwise $v_1 v v_3 v' v_1$ is a C_4 in $[V_{i-1} \cup V_i]$, contradicting the choice of i .) Now (2.1) guarantees the existence of u in V_{i-2} such that $(v, u) \in E$. In view of (2.4), $(u, v_2), (u, v_4) \in E$. So $\{v_1, v_2, v_3, v_4, v, w\}$ is isomorphic to F_1 with correspondence $v_1 \leftrightarrow p_2, v_2 \leftrightarrow p_3, v_3 \leftrightarrow p_4, v_4 \leftrightarrow p_1, v \leftrightarrow p_5, u \leftrightarrow p_6$, a contradiction.

(2.6) There is no C_4 in $[V_i \cup V_{i+1} \cup V_{i+2}]$ for $i = 1, 2, \dots, k-2$.

Assume the contrary: some $[V_i \cup V_{i+1} \cup V_{i+2}]$ contains a C_4 . Let i be such smallest subscript and let $v_1 v_2 v_3 v_4 v_1$ be a C_4 in $[V_i \cup V_{i+1} \cup V_{i+2}]$. Then, by (2.5), we may assume $v_1 \in V_i, v_2, v_4 \in V_{i+1}$ and $v_3 \in V_{i+2}$. In view of (2.1), there exists $v_5 \in V_i$ such that $(v_2, v_5) \in E$. Thus $(v_4, v_5) \in E$ for otherwise $v_1 v_2 v_5 v_4 v_1$ would be a C_4 in $[V_i \cup V_{i+1}]$, contradicting (2.5). Now we consider two cases.

In case $i = 2$, (2.2) guarantees the existence of $v_6 \in V_{i+2}$ with $(v_6, v_2) \in E$. In view of (2.5), $(v_6, v_4) \in E$. By (2.4), $(w, v_6), (w, v_3) \in E$. Thus $\{w, v_1, \dots, v_6\}$ is isomorphic to F_2 with correspondence $v_1 \leftrightarrow p_1, v_2 \leftrightarrow p_4, v_3 \leftrightarrow p_3, v_4 \leftrightarrow p_6, v_5 \leftrightarrow p_7, v_6 \leftrightarrow p_5, w \leftrightarrow p_2$, a contradiction.

In case $i \geq 3$, it can be shown similarly that there exist $v_6 \in V_{i-1}$ and $v_7 \in V_{i-2}$ with $(v_1, v_6), (v_5, v_6), (v_6, v_3), (v_6, v_7), (v_7, v_2), (v_7, v_4) \in E$. Thus $\{v_1, \dots, v_7\}$ induces an F_2 with the correspondence $v_1 \leftrightarrow p_1, v_2 \leftrightarrow p_4, v_3 \leftrightarrow p_3, v_4 \leftrightarrow p_6, v_5 \leftrightarrow p_7, v_6 \leftrightarrow p_2, v_7 \leftrightarrow p_5$, a contradiction.

Now let us introduce a partial order $\prec$ on each V_i as follows. For any $u, v \in V_i$, define $u \prec v$ if $d(u, v) = 2$, in other words, there exists a directed path from u to v with length 2. Then $\prec$ is well defined. To justify it, note that at most one of $u \prec v$ and $v \prec u$ can occur, since otherwise u and v would be on a C_4 in one of $[V_{i-1} \cup V_i]$, $[V_i \cup V_{i+1}]$, and $[V_{i-1} \cup V_i \cup V_{i+1}]$, contradicting (2.5) or (2.6). Moreover, $v \prec v$ will never occur for any vertex v .

(2.7) For any $x, y, z \in V_i$, if $x \prec y$ and $y \prec z$, then $x \prec z$.

Suppose $x \prec y$ is defined by path xuy and $y \prec z$ defined by path yvz . Then $u \neq v$ since $(u, y), (y, v) \in E$. It follows that $(x, v) \in E$ for otherwise $xuyvx$ would be a C_4 in one of $[V_{i-1} \cup V_i]$, $[V_i \cup V_{i+1}]$, and $[V_{i-1} \cup V_i \cup V_{i+1}]$, a contradiction. Thus path xvz yields $x \prec z$.

Let us now extend the partial order $\prec$ to a linear order on the whole V_i as follows: for any two incomparable (according to $\prec$) vertices $u, v \in V_i$, assign an arbitrary order between u and v . Then $\prec$ is the desired linear order.

(2.8) For any arc $(u, v) \in E$, we have $(u, x) \in E$ whenever $v \prec x$ and $(x, v) \in E$ whenever $x \prec u$.

Otherwise, we have path xuv , implying $x \prec v$ in the former case and have path uvx , implying $u \prec x$ in the latter, a contradiction.

The proof is complete. $\square$

Corollary 2.1 *Let bipartite tournament $T = (V, E)$ and $\{V_1, V_2, \dots, V_k\}$, a partition of V , be as described in Lemma 2.1. If $uvxyu$ is a C_4 in T , then there exists a subscript i with $1 \leq i \leq k-3$ such that $u \in V_{i+3}$, $v \in V_{i+2}$, $x \in V_{i+1}$, and $y \in V_i$ (renaming u, v, x and y if necessary).*

Proof. Since the distance between any two vertices on cycle $uvxyu$ is at most three, it follows from the definition of V_i that u, v, x and y are located in at most four consecutive sets, say $V_i \cup V_{i+1} \cup V_{i+2} \cup V_{i+3}$. On the other hand, by (2.5) and (2.6) any three consecutive sets contain no C_4 . So the desired statement follows. $\square$

Lemma 2.2 *Let $T = (V, E)$ be a strongly connected bipartite tournament. Then either one of F_1 and F_2 in T or a partition $\{V_1, V_2, \dots, V_k\}$ of V as described in Lemma 2.1 can be found in time $O(|V|^2)$.*

Proof. First note that the vertex w and the partition $\{V_1, V_2, \dots, V_k\}$ can be determined by the breadth-first search in time $O(|V|^2)$.

In order to establish a linear order for each V_i , let us set $X_2 = V_3$, $X_k = V_{k-1}$, $X_i = V_{i-1} \cup V_{i+1}$ for $3 \leq i \leq k-1$. Then the order of each V_i will be determined based on X_i . Since V_1 is a singleton, its order is trivial. Suppose we have determined the order of $V_1 \cup V_2 \dots \cup V_{i-1}$, let us proceed to the order of V_i , where $i \geq 2$.

Set $V_{i,1} = V_i$ and $k = 1$. Let $P_i = \{V_{i,1}, V_{i,2}, \dots, V_{i,k}\}$ be an ordered partition of V_i with $V_{i,1} \prec V_{i,2} \prec \dots \prec V_{i,k}$. We scan the vertices in X_i successively; suppose $v \in X_i$ is the vertex in our consideration.

(2.9) If there exist $x \in V_{i,h}$ and $y \in V_{i,j}$ such that

- $h < j$,
- $x \in N^+(v)$, and
- $y \in N^-(v)$,

then there exists a C_4 containing v, x and y in one of $[V_{i-1} \cup V_i]$, $[V_i \cup V_{i+1}]$ and $[V_{i-1} \cup V_i \cup V_{i+1}]$. Recall the proof of (2.5) and (2.6), we can thus output an F_1 or F_2 in time $O(|V|)$, stop; else,

(2.10) If there exists h , $1 \leq h \leq k$, such that

- $V_{i,h} \cap N^-(v) \neq \emptyset$,
- $V_{i,h} \cap N^+(v) \neq \emptyset$,
- $V_{i,j} \subset N^-(v)$ for all $j < h$, and
- $V_{i,j} \subset N^+(v)$ for all $j > h$,

then set $V_{i,h_1} = V_{i,h} \cap N^-(v)$, $V_{i,h_2} = V_{i,h} \cap N^+(v)$. Replace $V_{i,h}$ by $\{V_{i,h_1}, V_{i,h_2}\}$ in P_i with $V_{i,h_1} \prec V_{i,h_2}$, and replace k by $k+1$. (Since $N^-(v) \cap V_i = V_{i,h_1} \cup_{j=1}^{h-1} V_{i,j}$ and $N^+(v) \cap V_i = V_{i,h_2} \cup_{j=h+1}^{\ell_i} V_{i,j}$, for each $x \in V_{i,h_1} \cup_{j=1}^{h-1} V_{i,j}$ and each $y \in V_{i,h_2} \cup_{j=h+1}^{\ell_i} V_{i,j}$, there is a directed path xvy . So $x \prec y$ and thus the original order of P_i is preserved after the replacement of $V_{i,h}$.) We then scan the next vertex in X_i and repeat the process until no vertex is unscanned.

Since each $v \in X_i$ can be scanned in time $O(|V|)$, the total time complexity is $\sum_{i=2}^k O(|V||X_i|) + O(|V|^2) = O(|V|^2)$. $\square$

3 Optimal Cycle Packings

Let $\mathcal{C}_4$ be the set of all C_4 's in T and let H be the incidence matrix of $\mathcal{C}_4$ whose rows and columns are indexed by $\mathcal{C}_4$ and V , respectively, such that $H_{C,v} = 1$ if $v \in C$ and 0 otherwise for each $c \in \mathcal{C}$ and $v \in V$.

In this section, the bipartite tournament $T = (V, E)$ is confined to one with no F_1 nor F_2 . Let us proceed to investigate the (fractional) C_4 -packing problem

$$\max\{y^T e_m \mid y^T H \leq w^T, y \geq 0\} \quad (1)$$

where $m = |\mathcal{C}_4|$, $n = |V|$, e_m is the all-one column vector of size m , and w is in $R_+^n = \{x \in R^n \mid x \geq 0\}$.

Without loss of generality, we may assume that T is strongly connected. Since T contains no F_1 nor F_2 , V admits a partition $\{V_1, V_2, \dots, V_k\}$ as described in Lemma 2.1. Let D denote the digraph obtained from T by removing all arcs from V_i to V_j for any $i < j$, let P_4 denote a (directed) path with 4 vertices in D , and let $\mathcal{P}_4$ be the set of all P_4 's in D . Then it follows from Corollary 2.1 that there is a one-to-one correspondence between $\mathcal{C}_4$ of T and $\mathcal{P}_4$ of D . Hence $\mathcal{C}_4$ and $\mathcal{P}_4$ have the same incidence matrix, and the C_4 -packing problem on T is equivalent to the P_4 -packing on D . For convenience, we consider the latter problem instead of the former.

Let $\prec$ be the linear order on each V_i as defined in Lemma 2.1. Recall that the order does not apply to any two vertices in distinct V_i 's. Let us now fill this gap and extend $\prec$ to the whole vertex-set V of D .

(3.1) Define $u \prec v$ for any $u \in V_i$ and $v \in V_j$ with $i < j$.

Note that if $v_1 v_2 v_3 v_4$ is a P_4 in D then, according to (3.1), we have $v_4 \prec v_3 \prec v_2 \prec v_1$. The order $\prec$ on V leads to a lexicographic order on $\mathcal{P}_4$ as follows.

(3.2) Let $Q_1 = u_1 u_2 u_3 u_4$ and $Q_2 = v_1 v_2 v_3 v_4$ be two P_4 's in D . Define $Q_1 \prec Q_2$ if $u_j \prec v_j$ for the *largest* subscript j with $u_j \neq v_j$.

Based on (3.2), we can further derive a lexicographic order on the packing polytope $\{y \in R_+^m \mid y^T H \leq w^T\}$. For this purpose, we assume that the rows of the incidence matrix H are arranged in the *increasing* order of P_4 's.

(3.3) Let $y = (y_1, y_2, \dots, y_m)^T$ and $y' = (y'_1, y'_2, \dots, y'_m)^T$ be two fractional P_4 -packings in D , that is, both y and y' are in $\{y \in R_+^m \mid y^T H \leq w^T\}$. Define $y \prec y'$ if $y_j < y'_j$ for the *smallest* subscript j with $y_j \neq y'_j$.

(3.4) Two directed paths $P = u_1 u_2 u_3 u_4$ and $P' = u'_1 u'_2 u'_3 u'_4$ are said to be *crossing* if $P \prec P'$ and if there exist two vertices u_i, u'_j , $1 \leq i, j \leq 4$, and some subscript ℓ such that $u'_j \prec u_i$ and $u_i, u'_j \in V_\ell$.

For the above crossing pair, in view of Corollary 2.1, P is contained in $[\cup_{h=s}^{s+3} V_h]$ for some $s \leq k-3$, P' is contained in $[\cup_{h=t}^{t+3} V_h]$ for some t with $s \leq t \leq s+3$, and each V_h contains at least one and at most two vertices of P and P' , where $s \leq h \leq t+3$. Let v_h and v'_h denote the vertices in $V_h \cap (V(P) \cup V(P'))$ with $v_h \preceq v'_h$ ($v_h = v'_h$ if V_h contains only one vertex of P and P'). Define

$$P \wedge P' = v_{s+3} v_{s+2} v_{s+1} v_s \quad \text{and} \quad P \vee P' = v'_{t+3} v'_{t+2} v'_{t+1} v'_t.$$

Then we have

$$(3.5) \quad P \wedge P', P \vee P' \in \mathcal{P}_4 \text{ and } P \wedge P' \prec P \prec P' \prec P \vee P'.$$

Indeed, for each subscript h between $s+1$ and $s+3$, if neither P nor P' contains (v_h, v_{h-1}) , then $v_h \prec v'_h$ and (v'_h, v_{h-1}) is an arc of P or P' . Thus by Lemma 2.1 (b), $(v_h, v_{h-1}) \in E$. It follows that $P \wedge P' \in \mathcal{P}_4$. Similarly, $P \vee P' \in \mathcal{P}_4$. Since $u'_j \in P \wedge P'$ and $u_i \in P \vee P'$, we have $P \wedge P' \prec P \prec P' \prec P \vee P'$.

Lemma 3.1 *Let $T = (V, E)$ be a bipartite tournament with vertex-weight $w \geq 0$ and with no subdigraph isomorphic to F_1 nor F_2 . Then the lexicographically largest packing $\hat{y} \in \{y \in R_+^m \mid y^T H \leq w^T\}$ is an optimum solution to (1).*

Proof. Assume the contrary: $\bar{y}^T e_m > \hat{y}^T e_m$ for any optimum solution $\bar{y}$ to (1). For convenience, choose $\bar{y}$ to be the lexicographically largest among all the optimum solutions to (1). Since $\hat{y}$ is the lexicographically largest, there exists a directed path $Q^* = v_4 v_3 v_2 v_1$ such that $\hat{y}(Q^*) > \bar{y}(Q^*)$ and $\hat{y}(Q) = \bar{y}(Q)$ for all $Q \in \mathcal{P}_4$ with $Q \prec Q^*$. Let i be the subscript with $v_j \in V_{i-1+j}$ for each $1 \leq j \leq 4$ hereafter.

(3.6) Set $\mathcal{R} = \{Q \in \mathcal{P}_4 \mid \bar{y}(Q) > 0, Q \succ Q^*, V(Q) \cap V(Q^*) \neq \emptyset\}$. Then $\mathcal{R} \neq \emptyset$. Otherwise, let $\tilde{y}$ be the vector obtained from $\bar{y}$ by replacing $\bar{y}(Q^*)$ with $\hat{y}(Q^*)$. Then $\tilde{y}^T H \leq w^T$ and $\tilde{y}^T e_m > \bar{y}^T e_m$, a contradiction.

(3.7) No $Q \in \mathcal{R}$ contains any vertex in $\{v \in V_{i-1+j} \mid v \prec v_j\}$ for any $1 \leq j \leq 4$.

Assume the contrary: some $Q' \in \mathcal{R}$ contains a vertex $v^* \in V_{i-1+j}$ with $v^* \prec v_j$. Then Q' and Q^* form a crossing pair (recall (3.4)). By (3.5), $Q' \wedge Q^* \in \mathcal{P}_4$ and $Q' \wedge Q^* \prec Q^*$. Set $\delta = \min\{\bar{y}(Q'), \hat{y}(Q^*)\}$ and define

$$\tilde{y}(Q) = \begin{cases} \hat{y}(Q) + \delta & \text{if } Q = Q' \wedge Q^*, \\ \hat{y}(Q) & \text{if } Q \prec Q^* \text{ and } Q \neq Q' \wedge Q^*, \\ 0 & \text{otherwise.} \end{cases}$$

Then $\tilde{y}$ is feasible to (1) with $\hat{y} \prec \tilde{y}$, contradicting the definition of $\hat{y}$.

(3.8) No path $Q' \in \mathcal{R}$ satisfies $V(Q') \cap V(Q^*) \supseteq V(Q) \cap V(Q^*)$ for all $Q \in \mathcal{R}$.

Otherwise, define

$$\tilde{y}(Q) = \begin{cases} \bar{y}(Q) + \delta & \text{if } Q = Q^*, \\ \bar{y}(Q) - \delta & \text{if } Q = Q', \\ \bar{y}(Q) & \text{otherwise,} \end{cases}$$

where $\delta = \min\{\bar{y}(Q'), \hat{y}(Q^*) - \bar{y}(Q^*)\}$. It is easy to see that $\tilde{y}$ is also an optimum solution to (1). Since $\bar{y} \prec \tilde{y}$, we reach a contradiction. Hence

(3.9) There exist two vertices $v_h, v_j \in Q^*$ and two paths $R, R' \in \mathcal{R}$ such that $v_h \in R, v_j \notin R, v_j \in R', v_h \notin R'$.

Let us show that R and R' are crossing. Since $Q^* \prec R$ and $Q^* \prec R'$, neither R nor R' contains any vertex in V_r for any $r < i$. Without loss of generality, we may assume $h < j$. Then R must contain a vertex $u \in V_{i-1+j}$. From (3.7) and $v_j \in V_{i-1+j}$, it follows that $v_j \prec u$. Thus $R' \prec R$ for otherwise they are crossing. Similarly, R' must contain a vertex $u' \in V_{i-1+h}$. By (3.7), we have $v_h \prec u'$, implying that R and R' are crossing. Now set $\delta = \min\{\bar{y}(R), \bar{y}(R')\}$ and define

$$\tilde{y}(R) = \begin{cases} \bar{y}(Q) + \delta & \text{if } Q = R \wedge R' \text{ or } R \vee R', \\ \bar{y}(Q) - \delta & \text{if } Q = R \text{ or } R', \\ \bar{y}(Q) & \text{otherwise.} \end{cases}$$

Then it is easy to verify that $\tilde{y} \geq 0$, $\tilde{y}^T H = \bar{y}^T H$, and $\tilde{y}^T e_m = \bar{y}^T e_m$. Hence $\tilde{y}$ is an optimum solution to (1) with $\bar{y} \prec \tilde{y}$, contradicting the choice of $\bar{y}$.

This completes the proof. $\square$

Lemma 3.2 *In addition to the hypothesis of Lemma 3.1, if the weight w is integral, then the lexicographically largest packing $\bar{y} \in \{y \in R_+^m \mid y^T H \leq w^T\}$ is integral.*

Proof. According to the statement of Lemma 3.1, the lexicographically largest packing is optimum to (1). Based on this observation, we can come up with a greedy algorithmic proof of the present statement as follows.

At the current step, let $i \leq k-3$ be the smallest subscript with $V_i \neq \emptyset$ and let v_j^* be the smallest vertex in V_j with respect to the linear order $\prec$ as defined in Lemma 2.1. If $(v_{j+1}^*, v_j^*) \notin E$ for $j = i, i+1$, or $i+2$, then remove v_j^* from D ; else, set $Q^* = v_{i+3}^* v_{i+2}^* v_{i+1}^* v_i^*$, $y(Q^*) = \min\{w(v_j^*) \mid i \leq j \leq i+3\}$, $w(v_j^*) := w(v_j^*) - y(Q^*)$, $i \leq j \leq i+3$, and remove all the vertices v_j^* with $w(v_j^*) = 0$ from D , $i \leq j \leq i+3$. Repeat the process.

Since $y(Q^*)$ is integral for each iteration, the solution is integral. In view of Lemma 3.1, it is also optimum.

Recall Lemma 2.2, it takes $O(|V|^2)$ time to construct D . Since at least one vertex is removed from D in each iteration of the present algorithm, and each iteration takes time $O(|V|)$, the total time complexity of our algorithm is $O(|V|^2)$. $\square$

Combining Lemmas 3.1 and 3.2, we get the following result.

Theorem 3.1 *Let $T = (V, E)$ be a bipartite tournament with integral vertex-weight $w \geq 0$ and with no subdigraph isomorphic to F_1 nor F_2 . Then the lexicographically largest packing $\bar{y} \in \{y \in R_+^m \mid y^T H \leq w^T\}$ is an integral optimum solution to (1).*

4 The Min-Max Relation

Now let us turn to investigate the (fractional) C_4 -covering problem $\min\{w^T x \mid x \geq 0; Hx \geq e_m\}$, where $w \in Z_+^n$.

Theorem 4.1 *Let H be the $C_4 \times V$ incidence-matrix of a bipartite tournament T . Then the linear system $\{x \in R_+^n \mid Hx \geq e_m\}$ is TDI if and only if T contains no subdigraph isomorphic to F_1 nor F_2 .*

Proof. Recall the definition, $\{x \in R_+^n \mid Hx \geq e_m\}$ is TDI if and only if (1) has an integral optimal solution for any nonnegative integral vector w . The sufficiency follows directly from Theorem 3.1. Let us now justify the necessity. Suppose the contrary: T contains F_i , where $i = 1$ or 2 . Let w be such that $w(v) = 1$ if v is a vertex in F_i and 0 otherwise. Then (1) has no integral optimal solution with respect to w . To justify it, observe that

- (a) F_1 contains three C_4 's in total; each vertex in F_1 is on exactly two C_4 's.
- (b) F_2 contains six C_4 's in total; each vertex in F_2 is on at most five C_4 's.

Now set $y(C_4) = 1/2$ (resp. $1/5$) for each C_4 in F_1 (resp. in F_2) and 0 otherwise, then y is a feasible solution to (1) with $y^T e_m > 1$. Since for any integral solution $\tilde{y}$ to (1), $\tilde{y}^T e_m$ corresponds to the number of some vertex-disjoint C_4 's in F_i , which is at most 1. Thus (1) has no integral optimal solution with respect to w , a contradiction. $\square$

Edmonds and Giles [2] proved that if $\{Ax \geq b, x \geq 0\}$ is a TDI system, b is integral, and the minimum value in the LP-duality equation

$$\text{Min } \{c^T x \mid Ax \geq b, x \geq 0\} = \text{Max } \{y^T b \mid y^T A \leq c^T, y \geq 0\}$$

is finite for every integral vector c for which the maximum is finite, then the minimum has an integral optimum solution. Based on this theorem, Theorem 4.1 and Theorem 3.1, we can instantly establish the following min-max result.

Theorem 4.2 *Let $T = (V, E)$ be a bipartite tournament with integral vertex weight $w \geq 0$ and with no subdigraph isomorphic to F_1 nor F_2 . Then the C_4 -packing number of T equals the C_4 -covering number of T .*

Let T be a bipartite tournament with no F_1 nor F_2 . As shown in the proof of Lemma 3.2, a maximum C_4 -packing in T can be obtained by a greedy algorithm in time $O(|V|^2)$. Let us point out that a minimum C_4 -covering in T can be obtained in time $O(|V|^3)$ using the maximum C_4 -packing algorithm as a subroutine.

To start with, set $C = \emptyset$ and let D be the digraph as constructed in (3.1).

At the current step, let $i \leq k - 3$ be the smallest subscript with $V_i \neq \emptyset$ and let v_j^* be the smallest vertex in V_j with respect to the linear order $\prec$ as defined in Lemma 2.1. If $(v_{j+1}^*, v_j^*) \notin E$ for some $j = i, i + 1$, or $i + 2$, then remove v_j^* from D ; else, apply the maximum P_4 -packing algorithm to D and to each

$D - \{v_j^*\}$ for $i \leq j \leq i + 3$ to find optimum solutions, denoted by $\bar{y}(D)$ and $\bar{y}(D - \{v_j^*\})$, respectively. Let v_j^* , with $i \leq j \leq i + 3$, be a vertex satisfying $e^T \bar{y}(D - \{v_j^*\}) + w(v_j^*) = e^T \bar{y}(D)$. Set $C = C \cup \{v_j^*\}$ and $D = D - \{v_j^*\}$. Repeat the process.

Since it takes $O(|V|^2)$ to find the desired v_j^* by the maximum P_4 -packing algorithm and at least one vertex is removed at each iteration, the total time complexity of the algorithm is $O(|V|^3)$.

5 General Bipartite Tournaments

The min-max relation together with the above minimum cycle covering algorithm lead to a 3.5-approximation algorithm for the feedback vertex set problem on general bipartite tournaments, which relies on “eliminating” the problematic subdigraphs, F_1 and F_2 , from T .

Given a bipartite tournament $T = (V, E)$ such that each vertex $v \in V$ is associated with a positive integer $w(v)$, recall Lemma 2.2, we can find an F_1 or F_2 , or a partition $\{V_1, V_2, \dots, V_k\}$ of V as described in Lemma 2.1 in time $O(|V|^2)$. Set $C' = \emptyset$. If an F_j , where $j = 1$ or 2 , is output, then set $\delta = \min\{w(v) \mid v \in V(F_j)\}$, $w(v) = w(v) - \delta$ for all $v \in V(F_j)$, $C_0 = \{v \in V(F_j) \mid w(v) = 0\}$, $C' = C' \cup C_0$ and $T = T - C_0$; else, construct the digraph D as described in (3.1) and apply the minimum P_4 -covering algorithm to D to get a minimum P_4 -covering C'' for D . Then $C' \cup C''$ is a C_4 -covering of T . It is easy to see that the performance guarantee of the algorithm is 3.5.

On the other hand, the problem is NP-complete in general.

Theorem 5.1 *The feedback vertex set problem (given $c > 0$ whether there is a feedback vertex set of size at most c) on bipartite tournaments is NP-complete and approximable within 3.5.*

Proof. The approximation ratio follows from the above argument. Let us show the NP-completeness. Obviously, the problem is in NP. To prove the assertion, it suffices to reduce the 3-SATISFIABILITY problem (3SAT) to the feedback vertex set problem on bipartite tournaments. Let $U = \{u_1, u_2, \dots, u_n\}$ be the set of variables and let $C = \{c_1, c_2, \dots, c_m\}$ be the set of clauses in an arbitrary instance of 3SAT. We aim to construct a bipartite tournament $T = (V, E)$ such that T has a feedback vertex set of size $n + 2m$ if and only if C is satisfiable.

The construction consists of several components: truth-setting components, satisfaction testing components, and membership components, which are augmented by some additional arcs so that the resulting digraph is a bipartite tournament.

- For each variable $u_i \in U$, there is a truth-setting component $T_i = (V_i, E_i)$ with $V_i = \{u_i, \bar{u}_i, a_i, a'_i\}$ and $E_i = \{(u_i, a_i), (a_i, \bar{u}_i), (\bar{u}_i, a'_i), (a'_i, u_i)\}$. Note that T_i is a directed cycle of length four.

- For each clause $c_j \in C$, there is a satisfaction testing component $S_j = (V'_j, E'_j)$ with

$$\begin{aligned} V'_j &= \{x_j^1, x_j^2, x_j^3, y_j^1, y_j^2, y_j^3\}, \\ E'_j &= \{(y_j^h, x_j^h) | h = 1, 2, 3\} \cup \{(x_j^h, y_j^k) | 1 \leq h \neq k \leq 3\}. \end{aligned}$$

Notice that S_j is isomorphic to the forbidden digraph F_1 with the correspondence $x_j^h \leftrightarrow p_{2h}, y_j^h \leftrightarrow p_{2h-1}$ for $h = 1, 2, 3$.

- For each clause $c_j \in C$, let z_j^1, z_j^2 , and z_j^3 denote the three literals in c_j . For each literal z_j^i , there is a membership component $M_j^i = (\hat{V}_j^i, \hat{E}_j^i)$ with $\hat{V}_j^i = \{p_j^i, q_j^i\}$ and $\hat{E}_j^i = \{(p_j^i, q_j^i)\}$.

The following two sets will form the bipartition of the desired bipartite tournament.

$$\begin{aligned} W &= \{u_i, \bar{u}_i \in V_i | 1 \leq i \leq n\} \cup \{q_j^i \in \hat{V}_j^i | 1 \leq i \leq 3, 1 \leq j \leq m\} \\ &\quad \cup \{y_j^i \in V'_j | 1 \leq i \leq 3, 1 \leq j \leq m\}, \\ B &= \{a_i, a'_i \in V_i | 1 \leq i \leq n\} \cup \{p_j^i \in \hat{V}_j^i | 1 \leq i \leq 3, 1 \leq j \leq m\} \\ &\quad \cup \{x_j^i \in V'_j | 1 \leq i \leq 3, 1 \leq j \leq m\}. \end{aligned}$$

For convenience, write

$$V^* = \cup_{i=1}^n V_i, \quad V' = \cup_{j=1}^m V'_j, \quad \hat{V} = \cup_{j=1}^m \cup_{i=1}^3 \hat{V}_j^i.$$

Now let us proceed to the construction of the remaining arc-set.

- First add arc-set

$$\begin{aligned} &\{(W \cap V_i, B \cap V_j), (B \cap V_i, W \cap V_j) | 1 \leq i < j \leq n\} \\ &\cup \{(W \cap V_i', B \cap V_j'), (B \cap V_i', W \cap V_j') | 1 \leq i < j \leq m\} \\ &\cup \{(p_j^i, q_k^h), (q_j^i, p_k^h) | 1 \leq j < k \leq m, 1 \leq i \leq 3, 1 \leq h \leq 3\} \\ &\cup \{(p_j^i, q_j^h), (q_j^i, p_j^h) | 1 \leq j \leq m, 1 \leq i < h \leq 3\}. \end{aligned}$$

- Then add arc-set

$$\begin{aligned} &\{(W \cap V^*, B \cap (V' \cup \hat{V})), (B \cap (V' \cup \hat{V}), W \cap V^*)\} \\ &\cup \{(W \cap \hat{V}, B \cap V')\} \cup \{(B \cap \hat{V}, W \cap V')\}. \end{aligned}$$

- Finally, for each clause C_j and each literal $z_j^i \in C_j$, reverse the arc (z_j^i, x_j^i) , that is, if $z_j^i = u_k$ for some k , then replace (u_k, x_j^i) by (x_j^i, u_k) ; if $z_j^i = \bar{u}_k$ for some k , then replace $(\bar{u}_k, x_j^i)$ by $(x_j^i, \bar{u}_k)$.

The construction is completed. It is easy to see that the construction can be accomplished in polynomial time and the resulting digraph is a bipartite tournament with $12m + 4n$ vertices.

Let us show that T has a feedback vertex set of size $n + 2m$ if and only if C is satisfiable. Our proof heavily relies on the following observation: if B is a

feedback vertex set of T with $n + 2m$ vertices, then B^* contains exactly one vertex from each T_i and exactly two vertices from each S_j since T_i 's and S_j 's are pairwise vertex disjoint.

Sufficiency. Suppose T has a feedback vertex set of size $n + 2m$. Let B^* be such a feedback vertex set that minimizes $|B^* \cap \{a_i, a'_i \mid 1 \leq i \leq n\}|$. Then $B^* \cap \{a_i, a'_i \mid 1 \leq i \leq n\} = \emptyset$. To justify it, assume the contrary: $b_i \in B^*$ for some $b_i = a_i$ or a'_i . Set $B' = (B^* \setminus \{b_i\}) \cup \{u_i\}$. Then by the assumption on B^* , there exists a directed cycle C' with $C' \cap B' = \emptyset$. Thus $b_i \in C'$ and $u_i \notin C'$. Since $u_i a_i \bar{u}_i a'_i u_i$ is the unique directed cycle containing b_i in $T[V^* \cup \hat{V}]$, C' contains some vertex in V' . From the construction of T , it follows that C' contains an arc (x_j^h, z_j^h) , where $z_j^h \in c_j$ and $z_j^h = u_k$ or $\bar{u}_k$ for some k , whence $x_j^h z_j^h p_j^h q_j^h x_j^h$ is disjoint from B^* , contradicting the definition of B^* .

So $|B^* \cap \{u_i, \bar{u}_i\}| = 1$ for all $1 \leq i \leq n$, we can thus obtain a truth assignment $\tau : U \rightarrow \{\text{true}, \text{false}\}$ by setting $\tau(u_i) = \text{true}$ if $u_i \in B^*$ and $\tau(u_i) = \text{false}$ if $\bar{u}_i \in B^*$, $i = 1, 2, \dots, n$. It remains to show that each clause c_j is satisfied by τ . Indeed, for each satisfaction testing component S_j , at least one of x_j^1 , x_j^2 , and x_j^3 , say x_j^h , is outside B^* . So the literal $z_j^h \in B^*$ since B^* intersects the cycle $x_j^h z_j^h p_j^h q_j^h x_j^h$. Thus $\tau(z_j^h) = \text{true}$, in other words, c_j is satisfied.

Necessity. Suppose that $\tau : U \rightarrow \{\text{true}, \text{false}\}$ is a satisfying truth assignment for C . Then there exists at least one true literal in each clause. We choose one (denote it by $z_j^{h_j}$) from each clause c_j such that $\tau(z_j^{h_j}) = \text{true}$. Set

$$B^* = \{v_i \in \{u_i, \bar{u}_i\} \mid \tau(v_i) = \text{true}, 1 \leq i \leq n\} \cup \{x_j^i \neq x_j^{h_j} \mid 1 \leq i \leq 3; 1 \leq j \leq m\}.$$

Clearly, $|B^*| = n + 2m$. Moreover, each directed cycle in $T[V^* \cup \hat{V}]$ or in $T[V']$ is covered by B^* . Note that any other directed cycle contains some arc, say (x_j^i, z_j^i) , from V' to V^* . If $x_j^i \notin B^*$ then, by the definition of B^* , $\tau(z_j^i) = \text{true}$, implying $z_j^i \in B^*$. Thus B^* is a feedback vertex set. The proof is complete. $\square$

6 Concluding Remarks

We generalize the approach developed in our previous work [1] to the feedback vertex set problem on bipartite tournaments (which is shown to be NP-complete). The new structure characterization here is of its own interests and the proof is much simplified. The *TDI* characterization yields a 3.5-approximation algorithm for the feedback vertex set problem on general bipartite tournaments when combined with the subgraph removal technique. We are still interested in knowing whether this method of applying *TDI* characterization can be extended to wider range of combinatorial optimization problems and would like to pursue this direction further.

References

1. M. Cai, X. Deng, and W. Zang, A *TDI* System and Its Application to Approximation Algorithm, *Proc. 39th IEEE Symposium on Foundations of Computer Science*, Palo Alto, 1998, pp. 227-231.

2. J. Edmonds and R. Giles, A Min-max Relation for Submodular Functions on Graphs, *Annals of Discrete Mathematics* **1** (1977), 185-204.
3. J. Edmonds and R. Giles, Total Dual Integrality of Linear Systems, *Progress in Combinatorial Optimization* (ed. W. R. Pulleyblank), Academic Press, 1984, pp. 117-131.
4. R. Giles, and W.R. Pulleyblank, Total Dual Integrality and Integral Polyhedra, *Linear Algebra Appl.* **25** (1979), 191-196.
5. M. X. Goemans and D. P. Williamson, The Primal-Dual Method for Approximation Algorithms and Its Application to Network Design Problems, in: *Approximation Algorithms for NP-Hard Problems* (ed. D.S. Hochbaum), PWS Publishing Company, 1997, pp. 144-191.
6. A. Schrijver, Total Dual Integrality from Directed Graphs, Crossing Families and Sub- and Supermodular Functions, *Progress in Combinatorial Optimization* (ed. W. R. Pulleyblank), Academic Press, 1984, pp. 315-362.
7. A. Schrijver, Polyhedral Combinatorics, in *Handbook of Combinatorics* (eds. R.L. Graham, M. Grötschel, and L. Lovász), Elsevier Science B.V., Amsterdam, 1995, pp. 1649-1704.
8. A. Schrijver, On Total Dual Integrality, *Linear Algebra Appl.* **38** (1981), 27-32.
9. E. Speckenmeyer, On Feedback Problems in Digraphs, in: *Lecture Notes in Computer Science 411*, Springer-Verlag, 1989, pp. 218-231.

On the Separation of Maximally Violated mod- k Cuts

Alberto Caprara¹, Matteo Fischetti², and Adam N. Letchford³

¹ DEIS, University of Bologna,
viale Risorgimento 2, 40136 Bologna, Italy
acaprara@deis.unibo.it

² DEI, University of Padova,
via Gradenigo 6/A, 35131 Padova, Italy
fisch@dei.unipd.it

³ Dept. of Mgt. Science, Lancaster University,
Lancaster LA1 4YW, United Kingdom
a.n.letchford@lancaster.ac.uk

Abstract. Separation is of fundamental importance in cutting-plane based techniques for *Integer Linear Programming* (ILP). In recent decades, a considerable research effort has been devoted to the definition of effective separation procedures for families of well-structured cuts. In this paper we address the separation of Chvátal rank-1 inequalities in the context of general ILP's of the form $\min\{c^T x : Ax \leq b, x \text{ integer}\}$, where A is an $m \times n$ integer matrix and b an m -dimensional integer vector. In particular, for any given integer k we study *mod- k* cuts of the form $\lambda^T Ax \leq \lfloor \lambda^T b \rfloor$ for any $\lambda \in \{0, 1/k, \dots, (k-1)/k\}^m$ such that $\lambda^T A$ is integer. Following the line of research recently proposed for mod-2 cuts by Applegate, Bixby, Chvátal and Cook [1] and Fleischer and Tardos [16], we restrict to *maximally violated* cuts, i.e., to inequalities which are violated by $(k-1)/k$ by the given fractional point. We show that, for any given k , such a separation requires $O(mn \min\{m, n\})$ time. Applications to the TSP are discussed. In particular, for any given k , we propose an $O(|V|^2 |E^*|)$ -time exact separation algorithm for mod- k cuts which are maximally violated by a given fractional TSP solution with support graph $G^* = (V, E^*)$. This implies that we can identify a maximally violated TSP cut whenever a maximally violated (*extended*) *comb inequality* exists. Finally, specific classes of (sometimes new) facet-defining mod- k cuts for the TSP are analyzed.

1 Introduction

Separation is of fundamental importance in cutting-plane based techniques for *Integer Linear Programming* (ILP). In recent decades, a considerable research effort has been devoted to the definition of effective separation procedures for families of well-structured cuts. This line of research was originated by the pioneering work of Dantzig, Fulkerson and Johnson [12] on the *Traveling Salesman*

Problem (TSP) and led to the very successful branch-and-cut approach introduced by Padberg and Rinaldi [24]. Most of the known methods have been originally proposed for the TSP, a prototype in combinatorial optimization and integer programming.

In spite of the large research effort, however, polynomial-time exact separation procedures are known for only a few classes of facet-defining TSP cuts. In particular, no efficient separation procedure is known at present for the famous class of comb inequalities [19]. The only exact method is due to Carr [7], and requires $O(n^{2t+3})$ time for separation of comb inequalities with t teeth on a graph of n nodes. Recently, Letchford [21] proposed an $O(|V|^3)$ -time separation procedure for a superclass of comb inequalities, applicable when the fractional point to be separated has a planar support.

Applegate, Bixby, Chvátal and Cook [1] recently suggested concentrating on maximally violated combs, i.e., on comb inequalities which are violated by $1/2$ by the given fractional point x^* to be separated. This is motivated by the fact that maximally violated combs exhibit a very strong combinatorial structure, which can be exploited for separation. Their approach is heuristic in nature, and is based on the solution of a suitably-defined system of mod-2 congruences. Following this approach, Fleischer and Tardos [16] were able to design an $O(|V|^2 \log |V|)$ -time exact separation procedure for maximally violated comb inequalities for the case where the support graph $G^* = (V, E^*)$ of the fractional point x^* is planar.

It is well known that comb inequalities can be obtained by adding-up and rounding a convenient set of TSP degree equations and subtour elimination constraints weighed by $1/2$, i.e., they are $\{0, \frac{1}{2}\}$ -cuts in the terminology of Caprara and Fischetti [6]. These authors studied $\{0, \frac{1}{2}\}$ -cuts in the context of general ILP's. They showed that the associated separation problem is equivalent to the problem of finding a minimum-weight member of a binary clutter, i.e., a minimum-weight $\{0, 1\}$ -vector satisfying a certain set of mod-2 congruences. This problem is NP-hard in general, as it subsumes the max-cut problem as a special case.

In this paper we address the separation of Chvátal rank-1 inequalities in the context of general ILP's of the form $\min\{c^T x : Ax \leq b, x \text{ integer}\}$, where A is an $m \times n$ integer matrix and b an m -dimensional integer vector. In particular, for any given integer k we study *mod- k* cuts of the form $\lambda^T A x \leq \lfloor \lambda^T b \rfloor$ for any $\lambda \in \{0, 1/k, \dots, (k-1)/k\}^m$ such that $\lambda^T A$ is integer. We show that, for any given k , separation of maximally violated mod- k cuts requires $O(mn \min\{m, n\})$ time as it is equivalent to finding a $\{0, 1, \dots, k-1\}$ -vector satisfying a certain set of mod- k congruences. We also discuss the separation of maximally violated mod- k cuts in the context of the TSP. In particular, we show how to separate efficiently maximally violated members of a family of cuts that properly contains comb inequalities. Interestingly, this family contains facet-inducing cuts which are not comb inequalities. We also show how to reduce from $O(|V|^2)$ to $O(|V|)$ the number of tight constraints to be considered in the mod- k congruence system, where $|V|$ is the number of nodes of the underlying graph. We investigate specific

classes of (sometimes new) mod- k facet-defining cuts for the TSP and then give some concluding comments.

2 Maximally Violated mod- k Cuts

Given an $m \times n$ integer matrix A and an m -dimensional integer vector b , let $P := \{x \in \mathbb{R}^n : Ax \leq b\}$, $P_I := \text{conv}\{x \in \mathbb{Z}^n : Ax \leq b\}$, and assume $P_I \neq P$. A *Chvátal-Gomory cut* is a valid inequality for P_I of the form $\lambda^T Ax \leq \lfloor \lambda^T b \rfloor$, where the *multiplier vector* $\lambda \in \mathbb{R}_+^m$ is such that $\lambda^T A \in \mathbb{Z}^n$, and $\lfloor \cdot \rfloor$ denotes lower integer part. In this paper we address cuts which can be obtained through multiplier vectors λ belonging to $\{0, 1/k, \dots, (k-1)/k\}^m$ for any given integer $k \geq 2$. We call them *mod- k cuts*, as their validity relies on mod- k rounding arguments. Note that mod-2 cuts are in fact the $\{0, \frac{1}{2}\}$ -cuts studied in Caprara and Fischetti [6].

Any Chvátal-Gomory cut is a mod- k cut for some integer $k > 0$, as it is well known that undominated Chvátal-Gomory cuts only arise for $\lambda \in [0, 1]^m$, since replacing any λ_i by its fractional part $\lambda_i - \lfloor \lambda_i \rfloor$ always leads to an equivalent or stronger cut. Moreover, λ can always be assumed to be rational, i.e., an integer $k > 0$ exists such that $k\lambda$ is integer. Indeed, for any given $\lambda \in \mathbb{R}_+^m$ with $\alpha^T := \lambda^T A \in \mathbb{Z}^n$ one can obtain an equivalent (or better) multiplier vector $\tilde{\lambda}$ by solving the linear program $\min\{\tilde{\lambda}^T b : \tilde{\lambda}^T A = \alpha^T, \tilde{\lambda} \geq 0\}$, whose basic solutions are of the form $\tilde{\lambda} = [B^{-1}\alpha, 0]$ for some basis B of A^T . Hence $\det(B) \cdot \tilde{\lambda}$ is integer, as claimed.

We are interested in the following separation problem, in its optimization version:

mod- k SEP: *Given $x^* \in P$, find $\lambda \in \{0, 1/k, \dots, (k-1)/k\}^m$ such that $\lambda^T A \in \mathbb{Z}^n$, and $\lfloor \lambda^T b \rfloor - \lambda^T Ax^*$ is a minimum.*

Following [6], this problem can equivalently be restated in terms of the integer multiplier vector $\mu := k\lambda \in \{0, 1, \dots, k-1\}^m$. For any given $z \in \mathbb{Z}$ and $k \in \mathbb{Z}_+$, let $z \bmod k := z - \lfloor z/k \rfloor k$. As is customary, notation $a \equiv b \pmod{k}$ stands for $a \bmod k = b \bmod k$. Given an integer matrix $Q = (q_{ij})$ and $k \in \mathbb{Z}_+$, let $\overline{Q} = (\overline{q}_{ij}) := Q \bmod k$ denote the *mod- k support* of Q , where $\overline{q}_{ij} := q_{ij} \bmod k$ for all i, j . Then, mod- k SEP is equivalent to the following optimization problem.

mod- k SEP: *Given $x^* \in P$ and the associated slack vector $s^* := b - Ax^* \geq 0$, solve*

$$\delta^* := \min (s^{*T} \mu - \theta) \quad (1)$$

subject to

$$\overline{A}^T \mu \equiv 0 \pmod{k} \quad (2)$$

$$\overline{b}^T \mu \equiv \theta \pmod{k} \quad (3)$$

$$\mu \in \{0, 1, \dots, k-1\}^m \quad (4)$$

$$\theta \in \{1, \dots, k-1\}. \quad (5)$$

By construction, $(s^*{}^T \mu - \theta)/k$ gives the slack of the mod- k cut $\lambda^T A x \leq \lfloor \lambda^T b \rfloor$ for $\lambda := \mu/k$, computed with respect to the given point x^* . Hence, there exists a mod- k cut violated by x^* if and only if the minimum δ^* in (1) is strictly less than 0. Observe that $s^* \geq 0$ and $\theta \leq k-1$ imply $\delta^* \geq 1-k$, i.e., no mod- k cut can be violated by more than $(k-1)/k$. This bound is attained for $\theta = k-1$, when the mod- k congruence system (2)–(4) has a solution μ with $\mu_i = 0$ whenever $s_i^* > 0$. In this case, the resulting mod- k cut is said to be *maximally violated*.

Even for $k = 2$, mod- k SEP is NP-hard as it is equivalent to finding a minimum-weight member of a binary clutter [6]. However, finding a *maximally violated* mod- k cut amounts to finding any feasible solution of the congruence system (2)–(4) after having fixed $\theta = k-1$ and having removed all the rows of $(\bar{A}, \bar{b})$ associated with a strictly positive slack s_i^* . For any k prime this solution, if any exists, can be found in $O(mn \min\{m, n\})$ time by standard Gaussian elimination in $GF(k)$.

For k nonprime $GF(k)$ is not a field, hence Gaussian elimination cannot be performed. On the other hand, there exists an $O(mn \min\{m, n\})$ -time algorithm to find, if any, a solution of the mod- k congruence system (2)–(4) even for k nonprime, provided a prime factorization of k is known; see, e.g., Cohen [11].

The above considerations lead to the following result.

Theorem 1. *For any given k , maximally violated mod- k cuts can be found in $O(mn \min\{m, n\})$ time, provided a prime factorization of k is known.*

It is worth noting that mod- k SEP with $\mu_i = 0$ whenever $s_i^* > 0$ can be solved efficiently by fixing θ to any value in $\{1, \dots, k-1\}$. We call the corresponding solutions of (2)–(5) *totally tight* mod- k cuts. The following theorem shows that, for k prime, the existence of a totally tight mod- k cut implies the existence of a maximally violated mod- k cut.

Theorem 2. *For any k prime, a maximally violated mod- k cut exists if and only if a totally tight mod- k cut exists.*

Proof. One direction is trivial, as a maximally violated mod- k cut is also a totally tight mod- k cut. Assume now that a totally tight mod- k cut exists, associated with a vector (μ, θ) satisfying (2)–(5) and such that $\mu_i = 0$ for all $s_i^* > 0$. If $\theta \neq k-1$ and k is prime, μ can always be scaled by a factor $w \in \{2, \dots, k-1\}$ such that $\bar{A}^T w \mu \equiv 0 \pmod{k}$ and $\bar{b}^T w \mu \equiv k-1 \pmod{k}$.

Note that Theorem 2 cannot be extended to the case of k nonprime.

Of course, not all maximally violated mod- k cuts are guaranteed to be facet defining for P_I . In particular, a cut is not facet defining whenever it is associated with a nonminimal solution μ of the congruence system (2)–(4), where θ has been fixed to $k-1$ (barring the case of equivalent formulations of the same facet-defining cut). Indeed, the inequality associated with any solution $\tilde{\mu} \leq \mu$ is

violated whenever the one associated with μ is. Hence one is motivated in finding maximally violated mod- k cuts which are associated with minimal solutions. This can be done with no extra computational effort for k prime since, for any fixed θ , all basic solutions to (2)–(4) are minimal by construction. Unfortunately, the algorithm for k nonprime does not guarantee finding a minimal solution. On the other hand, the following result holds.

Theorem 3. *If there exists a maximally violated mod- k cut for some k nonprime, a maximally violated mod- ℓ cut exists also for every ℓ which is a prime factor of k .*

Proof. First of all, observe that $Qy \equiv d \pmod{k}$ implies $Qy \equiv d \pmod{\ell}$ for each prime factor ℓ of k . Hence, given a solution (μ, θ) of (2)–(5) with $\theta = k - 1$, the vector $(\mu, \theta) \bmod \ell$ yields a totally tight mod- ℓ cut, as $\theta \bmod \ell = k - 1 \bmod \ell \neq 0$. The claim then follows from Theorem 2.

It is then natural to concentrate on the separation of maximally violated mod- k cuts for some k prime. For several important problems these cuts define facets of P_I , as shown for the TSP in Section 4.

3 Separation of Maximally Violated mod- k Cuts for the TSP

The TSP polytope is defined as the convex hull of the characteristic vectors of all the Hamiltonian cycles of a given complete undirected graph $G = (V, E)$. For any $S \subseteq V$, let $\delta(S)$ denote the set of the edges with exactly one end node in S , and $E(S)$ denote the set of the edges with both end nodes in S . Moreover, for any $A, B \in V$ we write $E(A : B)$ for $\delta(A) \cap \delta(B)$. As is customary, for singleton node sets we write v instead of $\{v\}$. For any real function $x : E \rightarrow \mathbb{R}$ and for any $Q \subseteq E$, let $x(Q) := \sum_{e \in Q} x_e$.

A widely-used TSP formulation is based on the following constraints, called *degree equations*, *subtour elimination constraints (SEC's)*, and *nonnegativity constraints*, respectively:

$$x(\delta(v)) = 2, \text{ for all } v \in V \quad (6)$$

$$x(E(S)) \leq |S| - 1, \text{ for all } S \subset V, |S| \geq 2 \quad (7)$$

$$-x_e \leq 0, \text{ for all } e \in E. \quad (8)$$

We next address the separation of maximally violated mod- k cuts that can be obtained from (6)–(8). Given a point $x^* \in \mathbb{R}^E$ satisfying (6)–(8), we call *tight* any node set S with $x^*(E(S)) = |S| - 1$. It is well known that only $O(|V|^2)$ tight sets exist, which can be represented by an $O(|V|)$ -sized data structure called *cactus tree* [13]. A cactus tree associated with x^* can be found efficiently in $O(|E^*||V| \log(|V|^2/|E^*|))$, where $E^* := \{e \in E : x_e^* > 0\}$ is the support of x^* ; see [15] and also [20]. Moreover, we next show that only $O(|V|)$ tight sets need be considered explicitly in the separation of maximally violated mod- k cuts.

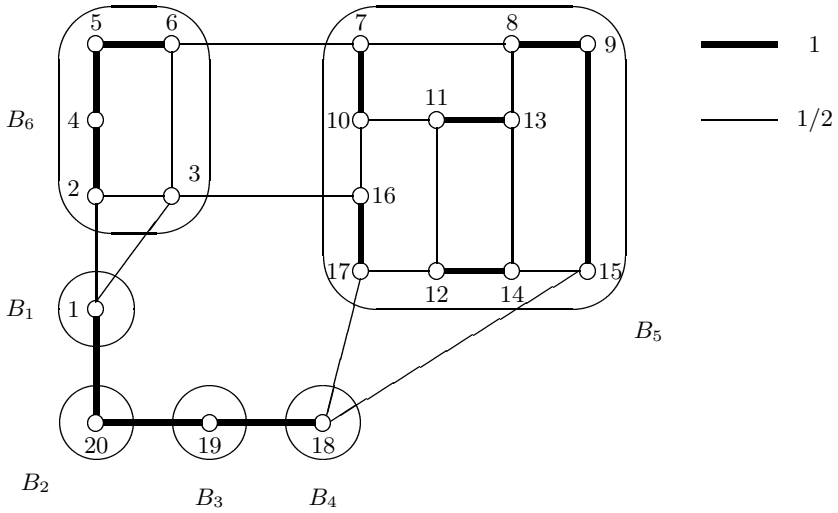


Fig. 1. A fractional point x^* and one of its necklaces.

Applegate, Bixby, Chvátal and Cook [1] and Fleischer and Tardos [16] showed that tight sets can be arranged in *necklaces*. A necklace of size $q \geq 3$ is a partition of V into a cyclic sequence of tight sets $B_1, \dots, B_q$ called *beads*; see Figure 1 for an illustration. To simplify notation, the subscripts in $B_1, \dots, B_q$ are intended modulo q , i.e., $B_i = B_{i+hq}$ for all integer h . Beads in a necklace satisfy:

- (i) $B_i \cup B_{i+1} \cup \dots \cup B_{i+t}$ is a tight set for all $i = 1, \dots, q$ and $t = 0, \dots, q-2$,
- (ii) $x^*(E(B_i : B_j))$ is equal to 1 if $j \in \{i+1, i-1\}$, and 0 otherwise.

A pair (B_i, B_{i+1}) of consecutive beads in a necklace is called a *domino*. We allow for degenerate necklaces with $q = 2$ beads, in which $x^*(E(B_1 : B_2)) = 2$. Degenerate necklaces have no dominoes.

Given x^* satisfying (6)–(8), one can find in time $O(|E^*||V| \log(|V|^2/|E^*|))$ a family $\mathcal{F}(x^*)$ of $O(|V|)$ necklaces with the property that every tight set is the union of consecutive beads in a necklace of the family. The next theorem shows that the columns in the congruence system (2)–(4) corresponding to tight SEC's are linearly dependent, in $GF(k)$, on a set of columns associated with degree equations, tight nonnegativity constraints, and tight SEC's corresponding to beads and dominoes in $\mathcal{F}(x^*)$.

Theorem 4. *If any TSP mod- k cut is maximally violated by x^* , then there exists a maximally violated mod- k cut whose Chvátal-Gomory derivation uses SEC's associated with beads and dominoes (only) of necklaces of $\mathcal{F}(x^*)$.*

Proof. Let S be any tight set whose SEC is used in the Chvátal-Gomory derivation of some maximally violated mod- k cut. By the properties of $\mathcal{F}(x^*)$, S is the union of consecutive beads $B_1, \dots, B_t$ of a certain necklace $B_1, \dots, B_q$ in $\mathcal{F}(x^*)$, $1 \leq t \leq q-1$. If $t \leq 2$, then S is either a bead or a domino, and there is nothing to prove. Assume then $t \geq 3$, as in Figure 2, and add together:

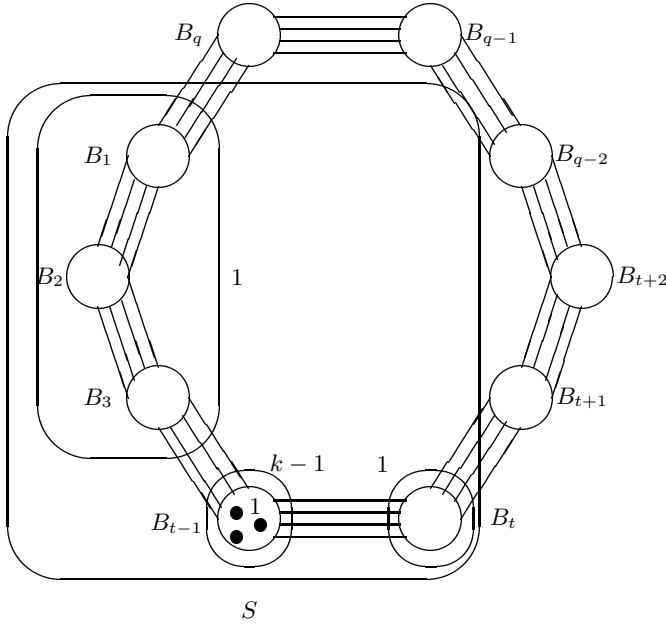


Fig. 2. Illustration for the proof of Theorem 4.

- the SEC on $B_1 \cup B_2 \cup \dots \cup B_{t-2}$,
- the SEC on B_{t-1} multiplied by $k - 1$,
- the SEC on B_t ,
- the degree equations on every $v \in B_{t-1}$,
- the nonnegativity inequalities $-x_e \leq 0$ for every $e \in E(B_{t-1} : B_{t+1} \cup \dots \cup B_q)$,
- the nonnegativity inequalities $-x_e \leq 0$ multiplied by $k - 1$ for every $e \in E(B_t : B_1 \cup \dots \cup B_{t-2})$.

This gives the following inequality:

$$\alpha^T x := x(E(S)) + kx(E(B_{t-1})) - kx(E(B_t : B_1 \cup \dots \cup B_{t-2})) \leq$$

$$\alpha_0 := |S| + k|B_{t-1}| - k - 1.$$

All the inequalities used in the combination are tight at x^* . Moreover, all the coefficients in $\alpha^T x \leq \alpha_0$ are identical, modulo k , to the coefficients of the SEC $x(E(S)) \leq |S| - 1$. So we can use the inequalities in the derivation of $\alpha^T x \leq \alpha_0$ in place of the original SEC to obtain a (different) maximally violated mod- k cut. Applying this procedure recursively yields the result.

As an immediate consequence, one has

Theorem 5. *For any given k , maximally violated mod- k cuts for the TSP can be found in $O(|E^*||V|^2)$ time, i.e., in $O(|V|^4)$ time in the worst case.*

Proof. Theorem 1 gives an $O(mn \min\{m, n\})$ -time separation algorithm, where m is the number of tight constraints (6)-(7), and $n = |E^*| = O(|V|^2)$ is the number of fractional components in x^* . By virtue of Theorem 4, only $O(|V|)$ tight sets need be considered explicitly, hence $m = O(|V|)$ and the claim follows.

The practical efficiency of the mod- k separation algorithm can be improved even further, as it turns out that one can always disregard all dominoes except one (arbitrarily chosen) in each necklace. This is expressed in the following theorem, whose proof is contained in the full paper.

Theorem 6. *Let $\mathcal{B}$ contain, for each necklace in $\mathcal{F}(x^*)$, all beads and a single (arbitrary) domino. If any TSP mod- k cut is maximally violated by x^* , then there exists a maximally violated mod- k cut whose Chvátal-Gomory derivation uses SEC's associated with sets $S \in \mathcal{B}$ only.*

In the full paper we will also derive similar results for the Asymmetric TSP polytope.

4 Specific Classes of mod- k Cuts for the TSP

In this section we analyze specific classes of facet-defining mod- k cuts for the Symmetric TSP. We also briefly mention some analogous results for the Asymmetric TSP which will be presented in detail in the full paper.

We first address mod-2 cuts that can be obtained from (6)–(8). A well known class of such cuts is that of *comb* inequalities, as introduced by Edmonds [14] in the context of matching theory, and extended by Chvátal [10] and by Grötschel and Padberg [17,18] for the TSP. Comb inequalities are defined as follows. We are given a *handle* set $H \subset V$ and $t \geq 3$, t odd, *tooth* sets $T_1, \dots, T_t \subset V$ such that $T_i \cap H \neq \emptyset$ and $T_i \setminus H \neq \emptyset$ hold for any $i = 1, \dots, t$. The comb inequality associated with $H, T_1, \dots, T_t$ reads:

$$x(E(H)) + \sum_{i=1}^t x(E(T_i)) \leq |H| + \sum_{i=1}^t (|T_i| - 1) - \frac{t+1}{2}. \quad (9)$$

The simplest case of comb inequalities arises for $|T_i| = 2$ for $i = 1, \dots, t$, leading to the Edmonds' *2-matching constraints*. It is well known that comb inequalities define facets of the TSP polytope [19]. Also well known is that comb inequalities are mod-2 cuts.

As already mentioned, no polynomial-time exact separation algorithm for comb inequalities is known at present. A heuristic scheme for maximally violated comb inequalities has been recently proposed by Applegate, Bixby, Chvátal and Cook [1], and elaborated by Fleischer and Tardos [16] to give a polynomial-time exact method for the case of x^* with planar support. Here, comb separation is viewed as the problem of “building-up” a comb structure starting with a given set of dominoes. The interested reader is referred to [1] and [16] for a detailed description of the method.

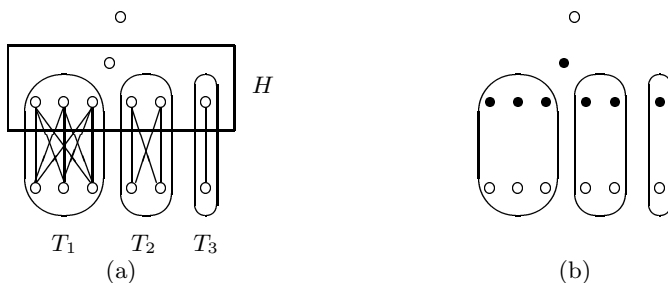


Fig. 3. (a) The support graph of a simple extended comb inequality; all the drawn edges, as well as the edges in $E(H)$, have coefficient 1. (b) A mod-2 derivation, obtained by combining the degree equations on the black nodes and the SEC's on the sets drawn in continuous line (the nonnegativity inequalities used in the derivation are not indicated).

Theorem 5 puts comb separation in a different light, in that it allows for efficient exact separation of maximally violated members of the family of mod-2 cuts which contains, among others, comb inequalities.

One may wonder whether comb inequalities are the only TSP facet-defining mod-2 cuts with respect to formulation (6)–(8). This is not the case; in particular, we address in the full paper the facet-defining *extended comb* inequalities of Naddef and Rinaldi [22] (see Figure 3 for an illustration) and prove the following

Theorem 7. *Extended comb inequalities are facet-defining TSP mod-2 cuts.*

Extended comb inequalities can be derived from 2-matching constraints by means of two general lifting operations, called *edge-cloning* and *0-node lifting*. These operations have been studied by Naddef and Rinaldi [23] who proved that, under mild assumptions, they preserve the facet-defining property of the original inequality. Interestingly, at least for the case of extended comb inequalities both operations do not increase the Chvátal rank [25] of the starting inequality, and also preserve the property of being a mod-2 cut. One may wonder whether this property is true in general. An answer to this question will be given in the full paper, where we study the two operations in the more general context of the Asymmetric TSP.

A family $\mathcal{N}$ of sets $S_1, \dots, S_k \subseteq V$ is called *nested* (or *laminar*) if, for all i, j , $S_i \cap S_j \neq \emptyset$ implies $S_i \subseteq S_j$ or $S_j \subseteq S_i$. The node sets associated with SEC's with nonzero multipliers in the Chvátal-Gomory derivation of an extended comb inequality define a nested family $\mathcal{N}$ with nesting degree not greater than 2, in the sense that $\mathcal{N}$ does not contain 3 subsets $S_1 \subset S_2 \subset S_3$. Actually, it is easy to show that any mod- k cut can be derived by only using SEC's associated with subsets defining a nested family. Interestingly, there are mod-2 facet-defining TSP cuts whose Chvátal-Gomory derivation involves SEC's with nesting level greater than

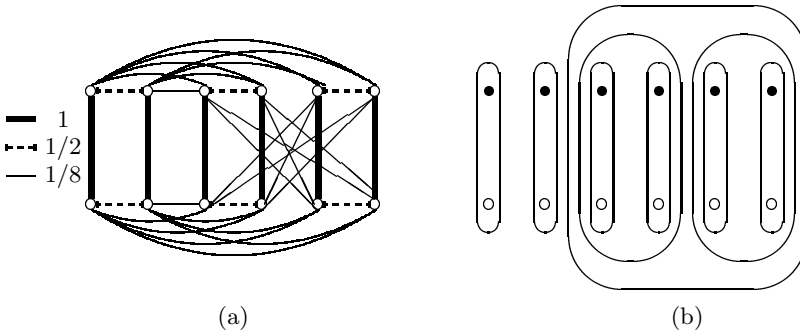


Fig. 4. (a) A fractional point x^* which violates maximally no extended comb inequality. (b) The derivation of a mod-2 cut which is maximally violated by x^* .

2. Here is an example. Consider the fractional point x^* of Figure 4(a). It is not hard to check by complete enumeration that x^* maximally violates no extended comb inequality. However, x^* maximally violates the mod-2 cut whose derivation is illustrated in Figure 4(b). It can be shown that this inequality is facet defining for the TSP polytope when $|V| \geq 12$.

Examples of facet-defining mod-3 cuts for the TSP are the Christof, Jünger and Reinelt [9] NEW1 inequality along with a generalization which we will introduce in the full paper.

Also in the full paper we give examples of facet-defining mod- k cuts for the Asymmetric TSP. These include, for $k = 2$, a generalization of the *source-destination* inequalities of Balas and Fischetti [5], for $k = 3$, generalizations of the NEW1 inequality and the *C3* inequalities of Grötschel and Padberg [19] and, finally, for arbitrary k , generalizations of the D_k^+ and D_k^- inequalities of Grötschel and Padberg [19].

5 Concluding Remarks

Recent developments in cutting-plane algorithms, such as the work of Balas, Ceria and Cornuéjols [2,3] and Balas, Ceria, Cornuéjols and Natraj [4] on lift-and-project (disjunctive) cuts and Gomory cuts, put the emphasis on the separation of large classes of inequalities which are not given explicitly. The approach developed in this paper provides still another tool for tackling hard problems.

Future theoretical research should be devoted to the study of the structure of undominated (facet-defining) mod- k TSP cuts. One should also address mod- k cuts for other combinatorial problems. Furthermore, the practical use of the separation methods herein proposed should be investigated.

References

1. D. Applegate, R. Bixby, V. Chvátal, W. Cook (1995). Finding cuts in the TSP (A preliminary report). Technical Report Technical Report 95-05, DIMACS, Rutgers University, New Brunswick, NJ.
2. E. Balas, S. Ceria, G. Cornuéjols (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program. (A)* 58, 295–324.
3. E. Balas, S. Ceria, G. Cornuéjols (1996). Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Sci.* 42, 1229–1246.
4. E. Balas, S. Ceria, G. Cornuéjols, N. Natraj (1996). Gomory cuts revisited. *Oper. Res. Lett.* 19, 1–9.
5. E. Balas, M. Fischetti (1993). A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets. *Math. Program. (A)* 58, 325–352.
6. A. Caprara, M. Fischetti (1996). $\{0, \frac{1}{2}\}$ -Chvátal-Gomory cuts. *Math. Program. (A)* 74, 221–235.
7. R. Carr (1995). Separating clique tree and bipartition inequalities in polynomial time. E. Balas, J. Clausen (eds.). *Integer Programming and Combinatorial Optimization 4*, Lecture Notes in Computer Science, 920, Berlin. Springer-Verlag, 40–49.
8. S. Ceria, G. Cornuéjols, M. Dawande (1995). Combining and strengthening Gomory cuts. E. Balas, J. Clausen (eds.). *Integer Programming and Combinatorial Optimization 4*, Lecture Notes in Computer Science, 920, Berlin. Springer-Verlag, 438–451.
9. T. Christof, M. Jünger, G. Reinelt (1991). A complete description of the traveling salesman polytope on 8 nodes. *Oper. Res. Lett.* 10, 497–500.
10. V. Chvátal (1973). Edmonds polytopes and weakly Hamiltonian graphs. *Math. Program.* 5, 29–40.
11. H. Cohen (1995). *A Course in Computational Algebraic Number Theory*, Springer-Verlag, Berlin.
12. G. Dantzig, D. Fulkerson, S. Johnson (1954). Solution of a large scale traveling-salesman problem. *Oper. Res.* 2, 393–410.
13. E.A. Dinitz, A.V. Karzanov, M.V. Lomonosov (1976). On the structure of a family of minimal weighted cuts in a graph. A.A. Fridman (ed.) *Studies in Discrete Optimization*, Moscow Nauka, 290–306 (in Russian).
14. J. Edmonds (1965). Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Natl. Bureau of Standards* 69, 125–130.
15. L. Fleischer (1998). Building the chain and cactus representations of all minimum cuts from Hao-Orlin in same asymptotic run time. R. Bixby, E. Boyd, R. Rios Mercado (eds.). *Integer Programming and Combinatorial Optimization 6*, Lecture Notes in Computer Science, Berlin. Springer-Verlag.
16. L. Fleischer, É. Tardos (1996). Separating maximally violated comb inequalities in planar graphs. W. Cunningham, S. McCormick, M. Queyranne (eds.). *Integer Programming and Combinatorial Optimization 5*, Lecture Notes in Computer Science, 1084, Berlin. Springer-Verlag, 475–489. Revised version to appear in *Math. Oper. Res.*
17. M. Grötschel, M. Padberg (1979). On the symmetric traveling salesman problem I: Inequalities. *Math. Program.* 16, 265–280.
18. M. Grötschel, M. Padberg (1979). On the symmetric traveling salesman problem II: lifting theorems and facets. *Math. Program.* 16, 281–302.

19. M. Grötschel, M. Padberg (1985). Polyhedral theory. E. Lawler, J. Lenstra, A. Rinnooy Kan, D. Shmoys (eds.). *The Traveling Salesman Problem*, John Wiley & Sons, Chichester, 251–305.
20. D. Karger, C. Stein (1996). A new approach to the minimum cut problem. *J. ACM* 43, 601–640.
21. A.N. Letchford (1998). Separating a superclass of comb inequalities in planar graphs. Technical Report, Dept. of Man. Science, The Management School, Lancaster University, 1998.
22. D. Naddef, G. Rinaldi (1988). The symmetric traveling salesman polytope: New facets from the graphical relaxation. Technical Report 248, IASI-CNR, Rome.
23. D. Naddef, G. Rinaldi (1993). The graphical relaxation: A new framework for the symmetric traveling salesman polytope. *Math. Program. (A)* 58, 53–88.
24. M. Padberg, G. Rinaldi (1991). A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* 33, 60–100.
25. A. Schrijver (1986). *Theory of Linear and Integer Programming*, John Wiley & Sons, New York.

Improved Approximation Algorithms for Capacitated Facility Location Problems

Fabián A. Chudak¹ and David P. Williamson²

¹ IBM T.J. Watson Research Center, Room 36-241, P.O. Box 218,
Yorktown Heights, NY, 10598. chudak@watson.ibm.com

² IBM T.J. Watson Research Center, Room 33-219, P.O. Box 218,
Yorktown Heights, NY, 10598. dpw@watson.ibm.com
<http://www.research.ibm.com/people/w/williamson>

Abstract. In a recent surprising result, Korupolu, Plaxton, and Rajaraman [10,11] showed that a simple local search heuristic for the capacitated facility location problem (CFLP) in which the service costs obey the triangle inequality produces a solution in polynomial time which is within a factor of $8 + \epsilon$ of the value of an optimal solution. By simplifying their analysis, we are able to show that the same heuristic produces a solution which is within a factor of $6(1 + \epsilon)$ of the value of an optimal solution. Our simplified analysis uses the supermodularity of the cost function of the problem and the integrality of the transshipment polyhedron.

Additionally, we consider the variant of the CFLP in which one may open multiple copies of any facility. Using ideas from the analysis of the local search heuristic, we show how to turn any α -approximation algorithm for this variant into one which, at an additional cost of twice the optimum of the standard CFLP, opens at most one additional copy of any facility. This allows us to transform a recent 3-approximation algorithm of Chudak and Shmoys [5] that opens many additional copies of facilities into a polynomial-time algorithm which only opens one additional copy and has cost no more than five times the value of the standard CFLP.

1 Introduction

We consider the *capacitated facility location problem* (CFLP). In this problem, we are given a set of facilities F and a set of clients D . Each client $j \in D$ has a demand d_j that must be serviced by one or more open facilities. There is a cost f_i for opening facility $i \in F$, and it costs c_{ij} for facility i to service one unit of demand from client j . We call the first type of cost *facility cost* and the second *service cost*. Furthermore, no facility may service more than U units of demand. We wish to service all clients at minimum total cost. The capacitated facility location problem and variations of it have been well-studied in the literature (see, for example, the book of Mirchandani and Francis [14]) and arise in practice (see, for example, the paper of Barahona and Jensen [3] for an instance of a parts warehousing problem from IBM).

The CFLP is NP-hard even in the case that $U = \infty$, sometimes called the *uncapacitated facility location problem* (UFLP) [7]. Thus we turn our attention to *approximation algorithms*. We say we have an α -approximation algorithm for the CFLP if the algorithm runs in polynomial time and returns a solution of value no more than α times the value of an optimal solution. The value α is sometimes called the *performance guarantee* of the algorithm.

It is possible to express any instance of the well-known set cover problem as an instance of the UFLP of the same cost, which implies that unless $P = NP$, there is no approximation algorithm for the UFLP with performance guarantee better than $c \ln |D|$, where c is some constant [13,8,16,1]. Thus we turn to special cases of the CFLP. In particular, we assume that for any $k, l \in F \cup D$ a service cost c_{kl} is defined, and the service costs are symmetric and obey the triangle inequality. This is a natural assumption, since service costs are often associated with the distance between points in Euclidean space representing facilities and clients. From now on, when we refer to the CFLP or UFLP, we refer to this metric case.

Recently, Korupolu, Plaxton, and Rajaraman (KPR) gave the first approximation algorithm for the CFLP with constant performance guarantee [10,11]. Surprisingly, Korupolu et al. show that a simple local search heuristic is guaranteed to run in polynomial time and to terminate with a solution of value no more than $(8 + \epsilon)$ times optimum, for any $\epsilon > 0$. The central contribution of our paper is to simplify and improve their analysis of the heuristic, showing that it is a $6(1 + \epsilon)$ -approximation algorithm for the CFLP. Although our proof follows theirs closely at many points, we show that some case distinctions (e.g. “cheap” versus “expensive” facilities) are unnecessary and some proofs can be simplified and strengthened by using standard tools from mathematical programming. For example, using the supermodularity of the cost function of the CFLP reduces a six and a half page proof to a half page, and using the notion of a transshipment problem and the integrality of its polyhedron allows us to get rid of the extraneous concept of a “refined β -allocation,” which in turn leads to the improved performance guarantee.

We are also able to use a concept translated from KPR to get an improved approximation algorithm for a variant of the CFLP. The variant we consider is the one in which a solution may open up to k copies of facility i , each at cost f_i and having capacity U , and we denote this problem the k -CFLP (so that the ordinary CFLP is the same as the 1-CFLP). Shmoys, Tardos, and Aardal [17] give a polynomial-time algorithm for the $\frac{7}{2}$ -CFLP which produces a solution of value no more than 7 times the optimal value of the 1-CFLP. Chudak and Shmoys [6], building on previous work [4,5] for the UFLP, give a 3-approximation algorithm for the ∞ -CFLP. Here we show how to take any solution for the ∞ -CFLP and produce from it a solution for the 2-CFLP adding cost no more than twice the optimal value of the 1-CFLP. Thus by using the Chudak-Shmoys algorithm, we are able to produce solutions in polynomial time for the 2-CFLP of cost no more than 5 times the optimal value of the 1-CFLP, improving the previous result of Shmoys et al. [17].

The recent work on approximation algorithms for facility location problems was started by the paper of Shmoys, Tardos, and Aardal [17], who gave a 3.16-approximation algorithm for the UFLP, the first approximation algorithm for this problem with a constant performance guarantee. Guha and Khuller [9] then gave a 2.41-approximation algorithm for the UFLP, which was followed by a 1.74-approximation algorithm due to Chudak and Shmoys [4,5]. Unlike the local search algorithm of KPR, the algorithms of these papers are based on techniques for deterministically and randomly rounding solutions to linear programming relaxations of the UFLP. Additionally, an observation of Sviridenko [18], combined with a result of Guha and Khuller [9] implies that no approximation algorithm for the UFLP with performance guarantee 1.46 is possible, unless $P = NP$.

The rest of the paper is structured as follows. We begin in Section 2, where we introduce the local search algorithm of KPR, define some notation, and prove some preliminary lemmas. We then, in Section 3, define the concept of a “swap graph”, analogous to the concept of the β -allocation problem in KPR, and show how it leads to our algorithm for the 2-CFLP. Finally, we show how to obtain an improved analysis of the local search algorithm using the swap graph in Section 4.

2 The Local Search Algorithm

2.1 Preliminaries

In this section, we define some notation and give some preliminary lemmas that will be needed in subsequent discussion. Given a set $S \subseteq F$ of facilities to open, it is easy to determine the minimum service costs for that set of facilities by solving the following transportation problem: for each facility $i \in F$ we have a supply node i with supply U , and for each client $j \in D$ we have a demand node j with demand d_j ; the unit shipping cost from i to j is c_{ij} . When discussing the k -CFLP, we will let S be a multiset of facilities (here l copies of $i \in F$ in S corresponds to opening l facilities at location i). We let $x(S, i, j)$ denote the amount of demand of client j serviced by facility i in the solution given by S . We will denote the overall cost of the location problem given by opening the facilities in S by $c(S)$. Furthermore, we let $c_f(S)$ denote the facility costs of the solution S (i.e., $c_f(S) = \sum_{i \in S} f_i$) and $c_s(S)$ denote the service costs of the solution S (i.e., $c_s(S) = \sum_{i \in F, j \in D} c_{ij} x(S, i, j)$). Let S^* denote the set of facilities opened by some optimal solution; it will always be a solution to the 1-CFLP and hence not a multiset. Let $n = |F|$.

The local search algorithm given by Korupolu et al. for the CFLP is the following: given a current solution S , perform any one of three types of operations that improve the value of the solution by at least $c(S)/p(n, \epsilon)$, where $p(n, \epsilon)$ is a suitably chosen polynomial in n and $1/\epsilon$, and continue doing so until none of these operations results in an improvement of at least that much. The operations are: *adding* a facility $i \in F - S$ to S (i.e., $S \leftarrow S + i$); *dropping* a facility $i \in S$ (i.e., $S \leftarrow S - i$); or *swapping* a facility $i \in S$ for a facility $i' \in F - S$ (i.e., $S \leftarrow S - i + i'$). We call any operation that improves the solution by at

least $c(S)/p(n, \epsilon)$ an *admissible* operation; thus the algorithm runs until there are no more admissible operations. This heuristic runs in polynomial time, as Korupolu et al. argued: start with some arbitrary feasible solution (for instance, setting $S = F$). Since in each step, the value of the solution improves by a factor of $(1 - \frac{1}{p(n, \epsilon)})$, after $p(n, \epsilon)$ operations the value of the solution will have improved by a constant factor. Since the value of the solution can't be smaller than $c(S^*)$, after $O(p(n, \epsilon) \log \frac{c(F)}{c(S^*)})$ operations the algorithm will terminate. Each local search step can be implemented in polynomial time, and $O(p(n, \epsilon) \log c(F))$ is a polynomial in the input size, so overall the algorithm takes polynomial time.

We now turn to proving some preliminary lemmas. These lemmas use the fact that the cost function c is supermodular; that is, if $A, B \subseteq F$, we have that

$$c(A) + c(B) \leq c(A \cap B) + c(A \cup B).$$

(See Babayev [2], Propositions 3.3 and 3.4 of Nemhauser, Wolsey, and Fisher [15].) In particular, c_s is supermodular, while c_f is modular (that is, $c_f(A) + c_f(B) = c_f(A \cap B) + c_f(A \cup B)$). We will use the fact that supermodularity holds even for multisets.

We show the following three lemmas:

Lemma 2.1. If $c(S) \geq (1 + \epsilon)c(S^*)$ and $S \subseteq S^*$, then there is an admissible add operation.

Lemma 2.2. If $c(S) \geq (1 + \epsilon)c(S^*)$ and $S \supseteq S^*$, then there is an admissible drop operation.

Lemma 2.3 (KPR [11], Lemma 9.3). If there is no admissible add operation, then $c_s(S) \leq c(S^*) + \frac{nc(S)}{p(n, \epsilon)}$.

In addition, in Section 4, we show the following theorem.

Theorem 2.4. If neither $S \subseteq S^*$ nor $S \supseteq S^*$ and there are no admissible drops or swaps, then

$$c_f(S - S^*) \leq 3c_f(S^* - S) + 2c_s(S) + 2c_s(S^*) + nc(S)/p(n, \epsilon).$$

2.2 The Main Theorem

Before proving the lemmas, we show how they lead to the $6(1 + \epsilon)$ -approximation algorithm for the CFLP.

Theorem 2.5. If there are no admissible operations, then

$$c(S) \leq 6(1 + \epsilon)c(S^*).$$

Proof. If there are no admissible operations and if $S \subseteq S^*$ or $S \supseteq S^*$, then by Lemmas 2.1 and 2.2 we know that $c(S) \leq (1+\epsilon)c(S^*)$. If there are no admissible operations and neither $S \subseteq S^*$ nor $S \supseteq S^*$ then

$$c_f(S - S^*) \leq 3c_f(S^* - S) + 2c_s(S) + 2c_s(S^*) + nc(S)/p(n, \epsilon),$$

by Theorem 2.4. Adding $c_f(S \cap S^*) + c_s(S)$ to both sides, we obtain

$$\begin{aligned} c(S) &\leq 2c_f(S^* - S) + c_f(S^*) + 3c_s(S) + 2c_s(S^*) + nc(S)/p(n, \epsilon) \\ &\leq 3c(S^*) + 3c(S^*) + 4nc(S)/p(n, \epsilon), \end{aligned}$$

using Lemma 2.3. Then

$$c(S) \left(1 - \frac{4n}{p(n, \epsilon)}\right) \leq 6c(S^*),$$

or

$$c(S) \leq \frac{6}{1 - \frac{4n}{p(n, \epsilon)}} c(S^*).$$

This gives that $c(S) \leq 6(1+\epsilon)c(S^*)$ for $p(n, \epsilon) \geq \frac{8n}{\epsilon}$ and $\epsilon < 1$. $\square$

2.3 Proofs of Preliminary Lemmas

We start by proving somewhat more general forms of Lemmas 2.1 and 2.2, and deriving those Lemmas as corollaries.

Lemma 2.6. Let $f : V \rightarrow \Re$ be any supermodular function. If $S \subset S^* \subseteq V$ and $f(S) \geq \gamma f(S^*)$, then there exists $u \in S^* - S$ such that

$$f(S + u) - f(S) \leq \frac{1}{|S^* - S|} \left(\frac{1}{\gamma} - 1 \right) f(S).$$

Proof. Let $W = S^* - S = \{u_1, u_2, \dots, u_k\}$. Let $W_i = \{u_1, \dots, u_i\}$. The statement certainly holds if $|S^* - S| = 1$, so assume that $|S^* - S| \geq 2$. Then by the supermodularity of f we know that:

$$\begin{aligned} f(S + W_{k-1}) + f(S + u_k) &\leq f(S^*) + f(S) \\ f(S + W_{k-2}) + f(S + u_{k-1}) &\leq f(S + W_{k-1}) + f(S) \\ &\vdots \\ f(S + W_2) + f(S + u_3) &\leq f(S + W_3) + f(S) \\ f(S + u_1) + f(S + u_2) &\leq f(S + W_2) + f(S). \end{aligned}$$

Summing the inequalities and subtracting $\sum_{i=2}^{k-1} f(S + W_i)$ from both sides, we obtain

$$\sum_{i=1}^k f(S + u_i) \leq f(S^*) + (k-1)f(S),$$

so that there exists some i such that

$$\begin{aligned} f(S + u_i) - f(S) &\leq \frac{1}{k}(f(S^*) - f(S)) \\ &\leq \frac{1}{k} \left(\frac{1}{\gamma} - 1 \right) f(S). \end{aligned}$$

□

Proof of Lemma 2.1: It follows from Lemma 2.6 that if $c(S) \geq (1 + \epsilon)c(S^*)$, then there exists an add operation that changes the cost by no more than $\frac{1}{n} \left(\frac{1}{1+\epsilon} - 1 \right) c(S) \leq -c(S)/p(n, \epsilon)$, for $p(n, \epsilon) \geq \frac{n(1+\epsilon)}{\epsilon}$. So there is an admissible add operation. □

Proof of Lemma 2.3: By modifying the last few lines of the proof of Lemma 2.6, it follows that if $f(S^*) \leq f(S) - \beta$, then there exists a $u_i \in S^* - S$ such that $f(S + u_i) - f(S) \leq -\frac{\beta}{|S^* - S|}$. Suppose it is the case that $c_s(S) > c(S^*) + \frac{nc(S)}{p(n, \epsilon)}$. Then by adding $c_f(S)$ to the left-hand side, $c_f(S - S^*)$ to the right-hand side, and observing that $c_s(S \cup S^*) \leq c_s(S^*)$, we have that $c(S) > c(S^* \cup S) + \frac{nc(S)}{p(n, \epsilon)}$. Setting $\beta = \frac{nc(S)}{p(n, \epsilon)}$ and applying the above gives us that there is an admissible add operation, proving the lemma. □

Lemma 2.7. Let $f : V \rightarrow \mathbb{R}$ be any supermodular function. If $S \supset S^*$ and $f(S) \geq \gamma f(S^*)$ for $\gamma \geq 1$, then there exists $u \in S - S^*$ such that

$$f(S - u) - f(S) \leq \frac{1}{|S - S^*|} \left(\frac{1}{\gamma} - 1 \right) f(S).$$

Proof. The proof of Lemma 2.7 is similar to that of Lemma 2.6, and so we omit it. □

Proof of Lemma 2.2: It follows from Lemma 2.7 that if $c(S) \geq (1 + \epsilon)c(S^*)$, then there exists a drop operation that changes the cost by no more than $\frac{1}{n} \left(\frac{1}{1+\epsilon} - 1 \right) c(S) \leq -c(S)/p(n, \epsilon)$, for $p(n, \epsilon) \geq \frac{n(1+\epsilon)}{\epsilon}$. So there is an admissible drop operation. □

Note then that we need $p(n, \epsilon) \geq \frac{n(1+\epsilon)}{\epsilon}$ (from the proofs of Lemmas 2.1 and Lemma 2.2) and $p(n, \epsilon) \geq \frac{8n}{\epsilon}$ (from the proof of Theorem 2.5). Thus $p(n, \epsilon) = \frac{8n}{\epsilon}$ is sufficient (assuming $\epsilon < 1$).

3 Path Decompositions and the Swap Graph

3.1 A Path Decomposition

In this section, we define a path decomposition and a concept called the swap graph which will be useful in both of our results. The path decomposition is more or less equivalent to the “difference graph” of Korupolu et al. [11] Appendix B, while the swap graph roughly corresponds to their “ β -allocation problem”. The

path decomposition is useful in comparing the value of our current solution with the optimal solution. The swap graph will be used in the analysis of the local search algorithm (in the proof of Theorem 2.4) and will be used in the algorithm and analysis of our result for the 2-CFLP.

To obtain the path decomposition, we start with some current solution S and the optimal solution S^* . We construct the following directed graph: we include a node j for each client $j \in D$, and a node i for each facility $i \in S \cup S^*$. We include an arc (j, i) of weight $w(j, i) = x(S^*, i, j)$ for all $i \in S^*, j \in D$ when $x(S^*, i, j) > 0$, and an arc (i, j) of weight $w(i, j) = x(S, i, j)$ for all $i \in S, j \in D$ when $x(S, i, j) > 0$. Observe that by the properties of x , the total weight of all arcs incoming to a node j for $j \in D$ is d_j , as is the total weight of all outgoing arcs. The total weight of arcs incoming to any node i for $i \in S^*$ is at most U , and the total weight of arcs going out of any node i for $i \in S$ is also at most U . Furthermore, notice that $\sum c_{ij}w(i, j) = c_s(S^*) + c_s(S)$.

By standard path-stripping arguments, we can decompose this graph into a set of paths $\mathcal{P}$ and cycles. We ignore the cycles; the paths start at nodes in S and end at nodes in S^* . Let the weight of a path P be denoted $w(P)$, and, overloading notation somewhat, let $c(P)$ denote its cost ($c(P) = \sum_{(i,j) \in P} c_{ij}$). Then $\sum_{P \in \mathcal{P}} c(P)w(P) \leq c_s(S) + c_s(S^*)$. For any subset of paths $\mathcal{P}' \subseteq \mathcal{P}$, let $\mathcal{P}'(A, \cdot)$ denote the set of paths in $\mathcal{P}'$ starting at nodes $i \in A$ for $A \subseteq S$, let $\mathcal{P}'(\cdot, B)$ denote the set of paths ending at nodes $i' \in B$ for $B \subseteq S^*$. Then $\mathcal{P}'(A, B)$ denotes the set of paths in $\mathcal{P}'$ from $i \in A \subseteq S$ to $i' \in B \subseteq S^*$. Also, let $w(\mathcal{P}') = \sum_{P \in \mathcal{P}'} w(P)$, and $val(\mathcal{P}') = \sum_{P \in \mathcal{P}'} w(P)c(P)$. Thus, for instance, $val(\mathcal{P}) \leq c_s(S) + c_s(S^*)$.

3.2 The Swap Graph

The swap graph simply corresponds to a transshipment problem from a specified subset S' of nodes of a current solution S (possibly a multiset) to a subset of facilities $F' \subset F$. We place demands of 1 on the nodes of S' and integer supplies on the facilities F' , and set the cost of an edge $\hat{c}_{kl}$ from $k \in S'$ to $l \in F - S$ to be $Uc_{kl} + f_l - f_k$. When using a swap graph, we use the path decomposition to prove that a fractional solution of some value β exists to the transshipment problem. Then by the integrality of the transshipment polyhedra, we know that there exists an integral solution to the transshipment problem of cost no more than β such that one unit of flow is shipped from each node in S' to exactly one node in F' .

We then observe that in the integral solution to the transshipment problem, each unit of flow from $k \in S'$ to $l \in F'$ corresponds to a swap operation in our current solution that can be performed while increasing the cost of the current solution by no more than $\hat{c}_{kl}$: each unit of demand assigned from client j to $k \in S'$ in the current solution can be assigned to $l \in F'$ at a change in cost of

$$c_{lj} - c_{kj} \leq c_{lk} + c_{kj} - c_{kj} \leq c_{kl}.$$

There are at most U units of demand assigned to $k \in S'$, so the total change in cost of transferring the demand assigned to k to l is at most Uc_{kl} , and the

change in cost of closing facility k and opening facility l is $f_l - f_k$. Thus the overall cost of performing the swap is at most $Uc_{kl} + f_l - f_k = \hat{c}_{kl}$.

3.3 An Algorithm for the 2-CFLP

To illustrate the use of the swap graph, we give an algorithm such that given any solution to the k -CFLP, for $k > 2$, the algorithm returns a solution to the 2-CFLP at additional cost no more than twice the cost of an optimal solution to the 1-CFLP. Given a solution S to the k -CFLP (a multiset), the algorithm works as follows. As long as there exists an add operation that reduces the cost, we add facilities from $F - S$ to S one at a time. Let $\tilde{S}$ be the solution when there are no longer add operations that improve the cost of the solution. We then solve a transshipment problem via the swap graph between nodes in $S' \subseteq \tilde{S}$ and F , where S' consists of all the copies of facilities that are used at full capacity; let $\tilde{S}_1 = \tilde{S} - S'$ be the remainder copies of facilities in S . We put demands of 1 on the nodes in S' , and supplies of 1 on the nodes in F , so that we have the following problem:

$$\begin{aligned} \text{Min} \quad & \sum_{k \in S', l \in F} \hat{c}_{kl} x_{kl} \\ \text{subject to:} \quad & \sum_{l \in F} x_{kl} = 1 \quad \forall k \in S' \\ & \sum_{k \in S'} x_{kl} \leq 1 \quad \forall l \in F \\ & x_{kl} \geq 0 \quad \forall k \in S', l \in F. \end{aligned}$$

Given the integral solution x_{kl} to the transshipment problem, whenever $x_{kl} = 1$, we obtain a new solution $\hat{S}$ by swapping $k \in S'$ for $l \in F$ at change in cost $\hat{c}_{kl}$. It is easy to verify that in solution $\hat{S}$ we open at most 2 facilities for each $i \in F$, one possibly from the assignment problem, and one from $\tilde{S}_1$, so that we have a solution to the 2-CFLP.

Certainly the algorithm runs in polynomial time. We can now prove that this algorithm does not increase the cost of the original solution by much.

Theorem 3.1. The algorithm above, given a solution S to the k -CFLP, produces a solution $\hat{S}$ to the 2-CFLP at additional cost at most twice the optimal value of a solution to the 1-CFLP.

Proof. We start with the solution S and apply add operations, each of which does not increase the cost of the solution. Given the solution $\tilde{S}$ (after we have applied all add operations to S that improve the cost of the solution), let $\mathcal{P}$ be the path decomposition giving paths from facilities in $\tilde{S}$ to an optimal 1-CFLP solution S^* . We use the path decomposition to give a fractional solution to the transshipment problem of cost no more than

$$c_s(\tilde{S}) + c_s(S^*) - c_f(S') + c_f(S^*) \leq c_s(\tilde{S}) + c(S^*).$$

By a small variation of Lemma 2.3, we know that we must have $c_s(\tilde{S}) \leq c(S^*)$. Since the cost of the solution $\hat{S}$ obtained after swapping is at most the cost of $c(\tilde{S})$ plus the cost of the solution to the transshipment problem, we know that

$$c(\hat{S}) \leq c(\tilde{S}) + 2c(S^*).$$

To obtain a feasible fractional solution $\tilde{x}$ to the transshipment problem, we set $\tilde{x}_{kl}$ to be $1/U$ times the total weight of paths from $k \in S'$ to $l \in S^*$ (that is, $\tilde{x}_{kl} = w(\mathcal{P}(k, l))/U$). Clearly $\tilde{x}$ is a feasible solution for the transshipment problem, since the total weight of paths leaving any $k \in S'$ is U , and the total weight of paths entering any $l \in S^*$ is at most U . The cost of the solution $\tilde{x}$ is

$$\begin{aligned} \sum_{k \in S', l \in S^*} \hat{c}_{kl} \tilde{x}_{kl} &= \sum_{k \in S', l \in S^*} (U c_{kl} - f_k + f_l)(1/U) \sum_{P \in \mathcal{P}(k, l)} w(P) \\ &\leq \sum_{k \in S', l \in S^*} \sum_{P \in \mathcal{P}(k, l)} w(P) c_{kl} - c_f(S') + c_f(S^*) \\ &\leq \sum_{P \in \mathcal{P}} c(P) w(P) - c_f(S') + c_f(S^*) \\ &\leq c_s(\tilde{S}) + c_s(S^*) - c_f(S') + c_f(S^*), \end{aligned}$$

where the inequality $c_{kl} \leq c(P)$ follows from the triangle inequality. $\square$

Corollary 3.2. There is a polynomial-time algorithm that finds a solution to the 2-CFLP of cost at most 5 times the optimal value of a 1-CFLP solution.

Proof. We apply the 3-approximation algorithm of Chudak and Shmoys [5] for the ∞ -CFLP to obtain our initial solution S . Since the cost of the optimal solution for ∞ -CFLP is at most the cost of the optimal solution for the 1-CFLP, the corollary follows. $\square$

4 Analysis of the Local Search Algorithm

We now use the path decomposition and swap graph tools from the previous section to complete our analysis of the local search algorithm, and prove Theorem 2.4. The lemmas we derive below are roughly similar to those of Korupolu et al. [11]: Lemma 4.2 corresponds to their Claim 9.7, and Lemma 4.3 to their Claims 9.8 and 9.9. However, we do not need an analogue of their “refined β -allocation”, which gives us an improvement in the analysis in Lemma 4.2.

Let S be a solution meeting the conditions of Theorem 2.4, and let $\mathcal{P}$ be the path decomposition for S and an optimal solution S^* . We will be particularly interested in three subsets of paths from $\mathcal{P}$. The first set is the set of all paths from nodes in $S - S^*$ to $S \cap S^*$ (if any); we call these the *transfer paths* and denote them $\mathcal{T} = \mathcal{P}(S - S^*, S \cap S^*)$. The basic idea of these paths in the proof is that for any path $P \in \mathcal{T}$, we claim we can transfer $w(P)$ of the demand assigned

to the start node of the path to the end node of the path at a cost of $c(P)w(P)$ without violating the capacity constraints. We establish this claim later.

The next subset of paths of interest is the set of all paths from $S - S^*$ to $S^* - S$; we call these the *swap paths* and denote them $\mathcal{S} = \mathcal{P}(S - S^*, S^* - S)$. We use the swap paths to get a fractional feasible solution for a transshipment problem from $S - S^*$ to $S^* - S$ in the swap graph, and get an integral solution of swaps whose cost is a simple expression in terms of $c_s(S)$, $c_s(S^*)$, $c_f(S - S^*)$, and $c_f(S^* - S)$. Thus if no swap can be performed that improves the cost of the current solution by a certain amount, this implies a bound on $c_f(S - S^*)$.

This idea does not quite work as stated because the weight of swap paths from $i \in S - S^*$ could be quite small. Thus, as in Korupolu et al. [11], we split the nodes of $S - S^*$ into two types: *heavy nodes* H such that the weight of paths from any $i \in H$ to $S^* - S$ is at least $U/2$ (i.e., $H = \{i \in S - S^* : w(\mathcal{S}(i, \cdot)) \geq U/2\}$), and *light nodes* (all the rest: $L = S - S^* - H$). We will be able to set up a transshipment problem for the nodes in H , which will give us a bound on $c_f(H)$. To get a bound on $c_f(L)$, we will have to set up a transshipment problem in a different manner and use the observation that we can transfer the demand assigned from one light node to another light node without violating capacity constraints.

To build towards our proof of Theorem 2.4, we now formalize the statements above in a series of lemmas.

Lemma 4.1. Weight $w(\mathcal{T}(i, \cdot))$ of the demand assigned to facility i in the current assignment can be transferred to other nodes in S at a cost increase of at most $val(\mathcal{T}(i, \cdot))$.

Proof. To prove the lemma, consider a path $P \in \mathcal{T}(i, \cdot)$, with start node i and end node i' . We observe that the first edge (i, j) in path P corresponds to a demand $w(P)$ assigned to i by client j in the current assignment. We reassign this demand to $i' \in S \cap S^*$; the increase in cost is at most $(c_{i',j} - c_{i,j})w(P) \leq c(P)w(P)$ by the triangle inequality. We now must show that such a reassignment does not violate the capacity constraints at i' . To see this, observe that by the properties of path-stripping, the total weight of paths incoming to any node $i' \in S^* \cap S$ is the difference between the total weight of arcs coming into node i' and the total weight of arcs going out of node i' . Since the total weight of arcs coming into node i' corresponds to the total amount of demand assigned to i' by the optimal solution, and the total weight of arcs going out of node i' corresponds to the total amount of demand assigned to i' by the current solution, and the optimal solution must be feasible, we can increase the demand serviced by i' by this difference and still remain feasible. $\square$

Lemma 4.2. If there is not an admissible swap operation, then

$$c_f(H) \leq 2c_f(S^* - S) + 2val(\mathcal{S}(H, \cdot)) + val(\mathcal{T}(H, \cdot)) + |H|c(S)/p(n, \epsilon).$$

Proof. As suggested in the exposition, we set up a transshipment problem from H to $S^* - S$, as follows:

$$\begin{aligned}
 & \text{Min} \quad \sum_{k \in H, l \in S^* - S} \hat{c}_{kl} x_{kl} \\
 & \text{subject to:} \\
 & \quad \sum_{l \in S^* - S} x_{kl} = 1 \quad \forall k \in H \\
 & \quad \sum_{k \in H} x_{kl} \leq 2 \quad \forall l \in S^* - S \\
 & \quad x_{kl} \geq 0 \quad \forall k \in H, l \in S^* - S.
 \end{aligned}$$

We claim that we can give a fractional solution to the transshipment problem of cost no more than $2\text{val}(\mathcal{S}(H, \cdot)) + 2c_f(S^* - S) - c_f(H)$. Thus there exists an integral solution of no greater cost. Given an integral solution x , when $x_{kl} = 1$, we can swap facility $k \in H$ for $l \in S^* - S$ and transfer the demand $w(\mathcal{S}(k, \cdot))$ assigned to k at change in cost at most $\hat{c}_{kl}$. By Lemma 4.1, we can transfer the remaining demand $w(\mathcal{T}(k, \cdot))$ assigned to k to nodes in $S \cap S^*$ at change in cost at most $\text{val}(\mathcal{T}(k, \cdot))$. By the hypothesis of the lemma, we know that any swap for a facility results in a change in cost of at least $-c(S)/p(n, \epsilon)$. Summing over all swaps for $k \in H$ given by the solution to the transshipment problem, we have that

$$2\text{val}(\mathcal{S}(H, \cdot)) + 2c_f(S^* - S) - c_f(H) + \text{val}(\mathcal{T}(H, \cdot)) \geq -\frac{|H|c(S)}{p(n, \epsilon)}.$$

Rearranging terms gives us the lemma.

To complete the proof, we give a fractional solution $\tilde{x}$ for this transshipment problem by setting

$$\tilde{x}_{kl} = \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))}.$$

Certainly the constraints $\sum_{l \in S^* - S} \tilde{x}_{kl} = 1$ are obeyed for all $k \in H$. The constraints $\sum_{k \in H} \tilde{x}_{kl} \leq 2$ are also obeyed since

$$\sum_{k \in H} \tilde{x}_{kl} = \sum_{k \in H} \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \leq \sum_{k \in H} \frac{w(\mathcal{S}(k, l))}{U/2} \leq 2,$$

where the first inequality follows by the definition of H and the second since the total weight of paths adjacent to any node is at most U . The cost of this fractional solution is

$$\begin{aligned}
 \sum_{k \in H, l \in S^* - S} \hat{c}_{kl} \tilde{x}_{kl} &= \sum_{k \in H, l \in S^* - S} (U c_{kl} + f_l - f_k) \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \\
 &\leq \sum_{k \in H, l \in S^* - S} \left[(U c_{kl} + f_l) \frac{w(\mathcal{S}(k, l))}{U/2} - f_k \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \right]
 \end{aligned}$$

$$\begin{aligned}
&\leq \sum_{k \in H, l \in S^* - S} 2c_{kl}w(\mathcal{S}(k, l)) + 2c_f(S^* - S) - c_f(H) \\
&\leq 2val(\mathcal{S}(H, \cdot)) + 2c_f(S^* - S) - c_f(H).
\end{aligned}$$

□

Lemma 4.3 (KPR [11], Claims 9.8 and 9.9). If there are no admissible drop and swap operations, then

$$c_f(L) \leq c_f(S^* - S) + 2val(\mathcal{T}(L, \cdot)) + 2val(\mathcal{S}(L, \cdot)) + |L|c(S)/p(n, \epsilon).$$

Proof. The proof of this lemma is similar to the proof of the previous lemma, although here we will have to set up a transshipment problem to capture both swap and drop operations. One difficulty with translating the previous proof to this case is ensuring that one can find a feasible fractional solution such that each facility in $F - S$ is in no more than a small constant number swap/drop operations. We do this by choosing exactly one “primary” facility k in L that can be swapped for a facility l in $F - S$; i.e. $x_{kl} > 0$ for exactly one $k \in L$. We make a careful choice of this facility k so that any other facility i to which we might otherwise normally make a fractional assignment $x_{il} > 0$, we can drop i and reassign its demand to k , the primary facility of l , at not much more cost.

We do this by setting up a transshipment problem from L to $(F - S) \cup L$, in which we set cost $\hat{c}_{kl} = w(\mathcal{S}(k, \cdot))c_{kl} + f_l - f_k$ for $l \in F - S$, $\hat{c}_{kl} = w(\mathcal{S}(k, \cdot))(c_{kl} + \theta(l)) - f_k$ for $l \in L, l \neq k$, and $\hat{c}_{kk} = \infty$, where $\theta(l)$ for $l \in L$ is the cost per unit demand for making $U/2$ units of capacity available at node l , either via the unused capacity at l or transferring demand via the paths $\mathcal{T}(l, \cdot)$.¹ Note that since $l \in L$, $w(\mathcal{S}(l, \cdot)) \leq U/2$, and thus the unused capacity at node l plus $w(\mathcal{T}(l, \cdot))$ is at least $U/2$. Thus $\frac{U}{2}\theta(l) \leq val(\mathcal{T}(l, \cdot))$. The transshipment problem is then:

$$\text{Min} \quad \sum_{k \in L, l \in (F - S) \cup L} \hat{c}_{kl}x_{kl}$$

subject to:

$$\begin{aligned}
\sum_{l \in (F - S) \cup L} x_{kl} &= 1 & \forall k \in L \\
\sum_{k \in L} x_{kl} &\leq 1 & \forall l \in F - S \\
x_{kl} &\geq 0 & \forall k \in L, l \in (F - S) \cup L.
\end{aligned}$$

We claim that we can give a fractional solution to the transshipment problem of cost no greater than $2val(\mathcal{S}(L, \cdot)) + val(\mathcal{T}(L, \cdot)) - c_f(L) + c_f(S^* - S)$. Thus there exists an integral solution of no greater cost. Given an integral solution x , when $x_{kl} = 1$ for $k \in L, l \in F - S$, we can swap facility $k \in L$ for $l \in F - S$

¹ The same cost function, including the definition of θ , was used by Korupolu et al. [11].

and transfer the demand $w(\mathcal{S}(k, \cdot))$ assigned to k at change in cost at most $\hat{c}_{kl}$. By Lemma 4.1, we can transfer the remaining demand $w(\mathcal{T}(k, \cdot))$ assigned to k to nodes in $S \cap S^*$ at change in cost at most $\text{val}(\mathcal{T}(k, \cdot))$. When $x_{ki} = 1$ for $k \in L, i \in L, k \neq i$, we can drop facility k from S and transfer the demand $w(\mathcal{S}(k, \cdot))$ assigned to k to i at change in cost $\hat{c}_{ki} = w(\mathcal{S}(k, \cdot))(c_{ki} + \theta(i))$, as this cost covers transferring these units of demand to i and transferring the same amount of demand from i to nodes in $S \cap S^*$. By Lemma 4.1, we can transfer the remaining demand $w(\mathcal{T}(k, \cdot))$ assigned to k to nodes in $S \cap S^*$ at change in cost at most $\text{val}(\mathcal{T}(k, \cdot))$. By the hypothesis of the lemma, we know that any swap or drop of a facility results in a change in cost of at least $-c(S)/p(n, \epsilon)$. Summing over all swaps and drops for $k \in L$ given by the solution to the transshipment problem, we have that

$$2\text{val}(\mathcal{S}(L, \cdot)) + 2\text{val}(\mathcal{T}(L, \cdot)) - c_f(L) + c_f(S^* - S) \geq -\frac{|L|c(S)}{p(n, \epsilon)}.$$

Rearranging terms gives us the lemma.

To complete the proof, we give a fractional solution $\tilde{x}$ for this transshipment problem. For each $l \in S^* - S$ we find $k \in L$ that minimizes $c_{kl} + \theta(k)$ and designate k as the primary node $\pi(l)$ for l . We then set $\tilde{x}_{kl}$ as follows. For each $l \in S^* - S$, if k is the primary node for l , we set $\tilde{x}_{kl} = w(\mathcal{S}(k, l))/w(\mathcal{S}(k, \cdot))$, otherwise $\tilde{x}_{kl} = 0$. For each $i \in L$, we set

$$\tilde{x}_{ki} = \sum_{l \in S^* - S: i = \pi(l), k \neq \pi(l)} \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))}.$$

This solution is feasible since certainly $\sum_{l \in F} x_{kl} = 1$ for all $k \in L$. Also, since for at most one $k \in L$ is $\tilde{x}_{kl} > 0$ for $l \in F - S$, $\sum_{k \in L} x_{kl} \leq 1$. Observe that when $\tilde{x}_{ki} > 0$ for $k \in L, i = \pi(l), l \in S^* - S$, then

$$\begin{aligned} \hat{c}_{ki} &= w(\mathcal{S}(k, \cdot))(c_{ki} + \theta(i)) - f_k \\ &\leq w(\mathcal{S}(k, \cdot))(c_{kl} + c_{il} + \theta(i)) - f_k \\ &\leq w(\mathcal{S}(k, \cdot))(2c_{kl} + \theta(k)) - f_k, \end{aligned}$$

since $c_{il} + \theta(i) \leq c_{kl} + \theta(k)$ by the definition of primary nodes. Then the cost of this fractional solution is

$$\begin{aligned} &\sum_{k \in L, l \in F} \hat{c}_{kl} \tilde{x}_{kl} \\ &= \sum_{k \in L, l \in S^* - S, k = \pi(l)} \hat{c}_{kl} \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} + \sum_{k \in L, i \in L} \hat{c}_{ki} \sum_{l \in S^* - S: i = \pi(l), k \neq \pi(l)} \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \\ &\leq \sum_{k \in L, l \in S^* - S, k = \pi(l)} \hat{c}_{kl} \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} + \sum_{k \in L, l \in S^* - S, i = \pi(l), k \neq i} \hat{c}_{ki} \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \\ &\leq \sum_{k \in L, l \in S^* - S, k = \pi(l)} [w(\mathcal{S}(k, \cdot))c_{kl} + f_l - f_k] \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \end{aligned}$$

$$\begin{aligned}
& + \sum_{k \in L, l \in S^* - S, k \neq \pi(l)} [w(\mathcal{S}(k, \cdot))(2c_{kl} + \theta(k)) - f_k] \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \\
& \leq \sum_{k \in L, l \in S^* - S} [w(\mathcal{S}(k, \cdot))(2c_{kl} + \theta(k)) - f_k] \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} + c_f(S^* - S) \\
& \leq \sum_{k \in L, l \in S^* - S} 2c_{kl} w(\mathcal{S}(k, l)) + \sum_{k \in L} \text{val}(\mathcal{T}(k, \cdot)) - c_f(L) + c_f(S^* - S) \\
& \leq 2\text{val}(\mathcal{S}(L, \cdot)) + \text{val}(\mathcal{T}(L, \cdot)) - c_f(L) + c_f(S^* - S).
\end{aligned}$$

□

Combining Lemmas 4.2 and 4.3 gives Theorem 2.4.

References

1. S. Arora and M. Sudan. Improved low-degree testing and its applications. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 485–495, 1997.
2. Dj. A. Babayev. Comments on the note of Frieze. *Mathematical Programming* 7:249–252, 1974.
3. F. Barahona and D. Jensen. Plant location with minimum inventory. *Mathematical Programming* 83:101–111, 1998.
4. F. Chudak. Improved approximation algorithms for uncapacitated facility location. In *Proceedings of the 6th IPCO Conference*, pages 180–194, 1998.
5. F. Chudak and D.B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. In preparation.
6. F. Chudak and D.B. Shmoys. Improved approximation algorithms for a capacitated facility location problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 875–876, 1999.
7. G. Cornuéjols, G. Nemhauser, and L. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, pages 119–171. John Wiley and Sons, Inc., New York, 1990.
8. U. Feige. A threshold of $\ln n$ for approximating set-cover. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 314–318, 1996.
9. S. Guha and S. Khuller. Greedy strikes back: improved facility location algorithms. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.
10. M. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 1998.
11. M. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. Technical Report 98-30, DIMACS, June 1998. Available from dimacs.rutgers.edu/TechnicalReports/1998.html.
12. E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York, 1976.
13. C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *JACM* 41:960–981, 1994.

14. P. Mirchandani and R. Francis, eds. *Discrete Location Theory*. John Wiley and Sons, Inc., New York, 1990.
15. G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming* 14:265–294, 1978.
16. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 475–484, 1997.
17. D. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 265–274, 1997.
18. M. Sviridenko, July, 1998. Personal communication.

Optimal 3-Terminal Cuts and Linear Programming

William H. Cunningham¹ and Lawrence Tang²

¹ Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON, Canada, N2L 3G1

² Department of Mathematics, University of British Columbia, Vancouver, BC, Canada V6T 1Y8

Abstract. Given an undirected graph $G = (V, E)$ and three specified terminal nodes t_1, t_2, t_3 , a *3-cut* is a subset A of E such that no two terminals are in the same component of $G \setminus A$. If a non-negative edge weight c_e is specified for each $e \in E$, the *optimal 3-cut problem* is to find a 3-cut of minimum total weight. This problem is NP-hard, and in fact, is max-SNP-hard. An approximation algorithm having performance guarantee $\frac{7}{6}$ has recently been given by Călinescu, Karloff, and Rabani. It is based on a certain linear programming relaxation, for which it is shown that the optimal 3-cut has weight at most $\frac{7}{6}$ times the optimal LP value. It is proved here that $\frac{7}{6}$ can be improved to $\frac{12}{11}$, and that this is best possible. As a consequence, we obtain an approximation algorithm for the optimal 3-cut problem having performance guarantee $\frac{12}{11}$.

1 Introduction

Given an undirected graph $G = (V, E)$ and k specified terminal nodes $t_1, \dots, t_k$, a *k-cut* is a subset A of E such that no two terminals are in the same component of $G \setminus A$. If a non-negative edge-weight c_e is specified for each $e \in E$, the *optimal k-cut problem* is to find a k -cut of minimum total weight. This problem was shown by Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis [5] to be NP-hard for $k \geq 3$. (Of course, it is solvable in polynomial time if $k = 2$.) They also gave a simple polynomial-time algorithm having performance guarantee $\frac{2(k-1)}{k}$, that is, it is guaranteed to deliver a k -cut of weight at most $\frac{2(k-1)}{k}$ times the minimum weight of a k -cut. Later, in [6], the same authors showed that for $k \geq 3$ the problem is max-SNP-hard, which implies that, assuming $P \neq NP$, there exists a positive ε such that the problem has no polynomial-time approximation algorithm with performance guarantee $1 + \varepsilon$.

The present paper concentrates on the optimal 3-cut problem. From the above remarks, it follows that this problem is max-SNP-hard, and the approximation algorithm of [6] has a performance guarantee of $\frac{4}{3}$. Recently, Călinescu, Karloff, and Rabani [1] gave an approximation algorithm having a performance guarantee of $\frac{7}{6}$. We give a further improvement that is based on their approach.

Chopra and Rao [3] and Cunningham [4] investigated linear programming relaxations of the 3-cut problem, showing results on classes of facets and separation algorithms. Here are the two simplest relaxations. (By a *T-path* we mean the edge-set of a path joining two of the terminals. By a *wye* we mean the edge-set of a tree having exactly three nodes of degree one, each of which is a terminal. For a set A , a subset B of A , and a vector $z \in \mathbf{R}^A$, $z(B)$ denotes $\sum_{j \in B} z_j$.)

$$(LP1) \quad \begin{array}{ll} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \\ & x(P) \geq 1, P \text{ a } T\text{-path} \\ & x_e \geq 0, e \in E. \end{array}$$

$$(LP2) \quad \begin{array}{ll} \text{minimize} & \sum_{e \in E} c_e x_e \\ \text{subject to} & \\ & x(P) \geq 1, P \text{ a } T\text{-path} \\ & x(Y) \geq 2, Y \text{ a wye} \\ & x_e \geq 0, e \in E. \end{array}$$

It follows from some simple observations about shortest paths, and the equivalence of optimization and separation, that both problems can be solved in polynomial time. It was proved in [4] that the approximation algorithm of [5] delivers a 3-cut of value at most $\frac{4}{3}$ times the optimal value of (LP1). (In particular, the minimum weight of a 3-cut is at most $\frac{4}{3}$ times the optimal value of (LP1).) It was conjectured that the minimum weight of a 3-cut is at most $\frac{16}{15}$ times the optimal value (LP2). The examples in Figure 1 (from [4]) show that this conjecture, if true, is best possible. In both examples, the values of a feasible solution x of (LP2) are shown in the figure. The weights c_e are all 2 for the example on the left. For the one on the right they are 1 for the edges of the interior triangle, and 2 for the other edges. In both cases the minimum 3-cut value is 8, but the given feasible solution of (LP2) has value 7.5.

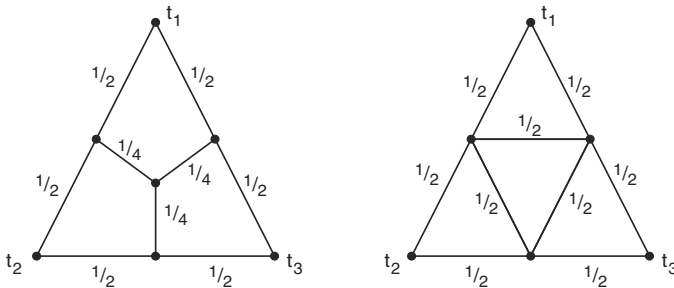


Fig. 1. Bad examples for (LP2)

Recently, Călinescu, Karloff, and Rabani [1] gave a new linear programming relaxation. Although their approach applies to any number k of terminals, we

continue to restrict attention to the case when $k = 3$. They need to assume that G be a complete graph. (Of course, if any missing edges are added with weight zero, the resulting 3-cut problem is equivalent to the given one, so this assumption is not limiting.) The relaxation is based on the following observations. First, every minimal 3-cut is of the form $\beta(R_1, R_2, R_3)$, where $t_i \in R_i$ for all i . Here, where $\mathcal{R}$ is a family of disjoint subsets of $\mathcal{R}$, $\beta(\mathcal{R})$ denotes the set of all edges of G joining nodes in different members of the family. Since $c \geq 0$, there is an optimal 3-cut of this form. Second, the incidence vector x of a minimal 3-cut is a kind of distance function, that is, it defines a function $d(v, w) = x_{vw}$ on pairs of nodes of G which is non-negative, symmetric, and satisfies the triangle inequality. Finally, with respect to d the distance between any two terminals is 1, and the sum of the distances from any node v to the terminals is 2. The resulting linear-programming relaxation is:

$$\begin{aligned}
 (LP3) \quad & \text{minimize} && \sum_{e \in E} c_e x_e \\
 & \text{subject to} && x_{vw} = 1, v, w \in T, v \neq w \\
 & && \sum_{v \in T} x_{vw} = 2, w \in V \\
 & && x_{uv} + x_{vw} - x_{uw} \geq 0, u, v, w \in V \\
 & && x_e \geq 0, e \in E.
 \end{aligned}$$

This relaxation is at least as tight as $(LP2)$. To see this, suppose that (after adding missing edges to make G complete), we have a feasible solution x to $(LP3)$. Then for any path P of G joining u to v , $x(P) \geq x_{uv}$, by applying the triangle inequality. It follows that $x(P) \geq 1$ for any T -path P . Moreover, any wye Y is the disjoint union of paths P_1, P_2, P_3 from some node v to the terminals. It follows that $x(Y) \geq \sum_{w \in T} x_{vw} = 2$. Thus every feasible solution of $(LP3)$ gives a feasible solution of $(LP2)$ having the same objective value. The first example of Figure 1 shows that the optimal value of $(LP3)$ can be strictly greater than the optimal value of $(LP2)$. On the other hand, the second example shows that there is no hope to prove in general that the minimum weight of a 3-cut is less than $\frac{16}{15}$ times the optimal value of $(LP3)$.

It was proved in [1] that the minimum weight of a 3-cut is at most $\frac{7}{6}$ times the optimal value of $(LP3)$. As a consequence, an approximation algorithm for the optimal 3-cut problem having a performance guarantee of $\frac{7}{6}$ was derived. (It is clear that $(LP3)$ can be solved in polynomial time, since it is of polynomial size.) However, it was left open whether this result could be strengthened; the second example of Figure 1 shows an example for which the minimum weight of a 3-cut can be as large as $16/15$ times the optimal value of $(LP3)$, and this is the worst example given in [1]. (To see that x of that example does extend to a feasible solution of $(LP3)$, we simply define x on each missing edge uv to be the minimum length, with respect to lengths x_e , of a path from u to v .)

In this paper it is shown that the minimum weight of a 3-cut is at most $\frac{12}{11}$ times the optimal value of $(LP3)$, and that this is best possible. (This result has been obtained independently by Karger, Klein, Stein, Thorup, and Young [7].) As a consequence we obtain an approximation algorithm for the optimal 3-cut problem having a performance guarantee of $\frac{12}{11}$.

2 Triangle Embeddings

Călinescu, Karloff, and Rabani [1] introduced an extremely useful geometric relaxation, which they showed was equivalent to the linear-programming relaxation (*LP3*). Let Δ denote the convex hull of the three elementary vectors $e^1 = (1, 0, 0)$, $e^2 = (0, 1, 0)$, and $e^3 = (0, 0, 1)$ in $\mathbf{R}^3$. By a *triangle embedding* of G we mean a mapping y from V into Δ such that $y(t_i) = e^i$ for $i = 1, 2, 3$. A triangle embedding y determines a vector $x \in \mathbf{R}^E$ as follows. For each edge uv , let x_{uv} be one-half the L_1 distance from $y(u)$ to $y(v)$. It is easy to see that this x is a feasible solution to (*LP3*). Conversely, a feasible solution x of (*LP3*) determines a triangle embedding y as follows. For each node v , let $y(v) = (1 - x_{t_1v}, 1 - x_{t_2v}, 1 - x_{t_3v})$.

Given a triangle embedding y we can obtain x as above, and then use x to obtain a triangle embedding y' . It is easy to see that $y = y'$. It is not true, on the other hand, that every feasible solution of (*LP3*) arises in this way from a triangle-embedding. However, it is “almost true”. The following result is implicit in [1], and we include a proof for completeness.

Theorem 1. *Let x be a feasible solution of (*LP3*), let y be the triangle embedding determined by x and let x' be the feasible solution of (*LP3*) determined by y . Then $x' \leq x$, and if x is an optimal solution of (*LP3*), so is x' .*

Proof. First, observe that the second statement is a consequence of the first and the non-negativity of c . Now let $uv \in E$. Both $y(u)$ and $y(v)$ have component-sum 1. Therefore, $y(u) - y(v)$ has component-sum zero, and so one-half of the L_1 distance between $y(u)$ and $y(v)$ is the sum of the non-negative components of $y(u) - y(v)$. Hence we may assume, perhaps by interchanging u with v and relabelling the terminals, that one-half of the L_1 distance between $y(u)$ and $y(v)$ is the sum of the first two components of $y(u) - y(v)$. Therefore,

$$\begin{aligned} \frac{1}{2} \|y(u) - y(v)\|_1 &= y_1(u) - y_1(v) + y_2(u) - y_2(v) \\ &= 1 - x_{ut_1} - (1 - x_{vt_1}) + 1 - x_{ut_2} - (1 - x_{vt_2}) \\ &= (2 - x_{ut_3}) - (2 - x_{vt_3}) \\ &\leq x_{uv}, \end{aligned}$$

as required. □

The approximation algorithm of Călinescu, Karloff, and Rabani uses the following ideas. Suppose that (*LP3*) is solved, and an optimal solution x^* that arises from a triangle embedding is found. For a number α between 0 and 1 that is different from x_{rv}^* for every $v \in V$ and $r \in T$, and an ordering r, s, t of T , define $R_r = \{v \in V : x_{rv}^* < \alpha\}$, $R_s = \{v \in V \setminus R_r : x_{sv}^* < \alpha\}$, $R_t = V \setminus (R_r \cup R_s)$. We call the 3-cut $\beta(R_r, R_s, R_t)$ *uniform* (with respect to this x^*). It is easy to see that there are $O(n)$ uniform 3-cuts. The algorithm of [1] simply chooses the uniform 3-cut having minimum weight. It is proved to have weight at most $\frac{7}{6}$ times the minimum weight of a 3-cut.

We consider a slight generalization of the notion of uniform 3-cut. Let α, α' be two numbers chosen as α was above, and let r, s, t be an ordering of T . Define $R_r = \{v \in V : x_{rv}^* < \alpha\}$, $R_s = \{v \in V \setminus R_r : x_{sv}^* < \alpha'\}$, $R_t = V \setminus (R_r \cup R_s)$. We call the 3-cut $\beta(R_r, R_s, R_t)$ *flat* (with respect to this x^*). Clearly, every uniform 3-cut is flat. It is easy to see that there are $O(n^2)$ flat 3-cuts. Our approximation algorithm simply chooses the flat 3-cut having minimum weight. We will show that it has weight at most $\frac{12}{11}$ times the weight of an optimal 3-cut. This result is based on a tight analysis of the bound for the optimal 3-cut problem given by (LP3).

3 Linear Programming Again

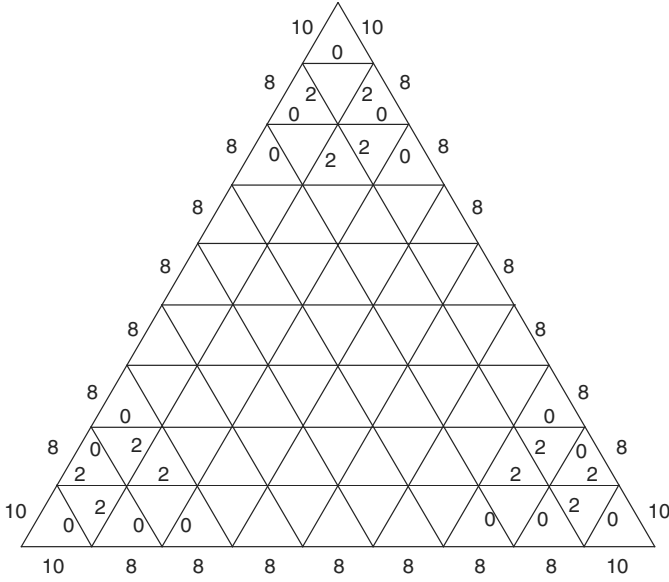
It is easy to check that if the optimal value of (LP3) is zero, then there is a 3-cut of weight zero. Therefore, we may assume that the optimal value is positive. So our problem may be restated as finding the best upper bound, over all choices of G and c , for the minimum weight of a 3-cut divided by the optimal value of (LP3). By multiplying c by an appropriate positive number, we may assume that the minimum weight of a 3-cut is 1. It is now more convenient to prove the best lower bound on the value of (LP3). Surprisingly, we can use a different linear programming problem to do this.

Assume that G is fixed, and that an optimal solution x^* of (LP3) is also fixed. Then the problem of finding the worst optimal value can be stated as:

$$(P) \quad \begin{array}{ll} \text{minimize} & \sum_{e \in E} c_e x_e^* \\ \text{subject to} & \\ & c(S) \geq 1, \quad S \text{ a 3-cut} \\ & c_e \geq 0, \quad e \in E. \end{array}$$

Note that the variables are the weights c_e ! It may seem that the hypothesis that G and x^* are known is very strong, but it turns out that we can assume that there are not many choices for them. First, we may assume that x^* is rational, since it is an optimal solution of a linear-programming problem having rational data. Therefore, there exists a positive integer q such that qx^* is integer-valued. Second, we may assume that x^* arises from a triangle-embedding y^* , and it is easy to see that qy^* is integral, as well. Therefore, we can think of y^* as embedding the nodes of G into a finite subset Δ_q of Δ , consisting of those points $y \in \Delta$ for which qy is integral. We define the planar graph $G_q = (\Delta_q, E_q)$ by $uv \in E_q$ if and only if the L_1 distance between u and v is $\frac{2}{q}$. Figure 3 shows G_9 ; the numbers there are explained later. For nodes u, v of G_q , we denote by $d_q(u, v)$ the least number of edges of a path in G_q from u to v . (It is easy to see that $d_q(u, v) = \frac{q}{2}$ times the L_1 distance from u to v .)

Theorem 2. *Let G, c be a 3-cut instance, let x^* be a rational-valued optimal solution of (LP3), with corresponding triangle-embedding y^* , and let q be a positive integer such that qx^* is integral. Then there is a 3-cut instance on graph $\hat{G}$ with nodeset Δ_q and edge-weights $\hat{c}$ such that:*

Fig. 2. G_9

- (a) $\hat{x}$ defined by $q\hat{x}_{uv} = d_q(u, v)$ for all $uv \in E$ is a feasible solution of (LP3) (for $\hat{G}, \hat{c}$), and $\hat{c}\hat{x} \leq cx^*$;
- (b) The optimal 3-cut value for $\hat{G}, \hat{c}$ is at least that for G, c ;
- (c) $\hat{c}_e = 0$ for all $e \notin E_q$;
- (d) For every flat 3-cut of $\hat{G}$ with respect to $\hat{x}$, there is a flat 3-cut of G with respect to x^* having no larger weight.

Proof. We use the mapping y^* from V to Δ_q , and we assume that x^* arises from y^* . Suppose that two nodes u, v of G are mapped to the same point of Δ_q by y^* . Form G' by identifying u with v and, where multiple edges are formed, replacing the pair by a single edge whose weight is their sum. Then every 3-cut of G' determines a 3-cut of G having the same weight, so the minimum weight of a 3-cut of G' is at least the minimum weight of a 3-cut of G . Moreover, x^* also determines a triangle-embedding of G' , so there is a feasible solution of (LP3) for G' having value cx^* . Finally, every flat cut of G' gives a flat cut of G of the same weight. Thus the theorem is true for G if it is true for G' , and so we may assume that y^* is one-to-one.

Now suppose that y^* is not onto, that is, that there is an element z of Δ_q such that $y^*(v) \neq z$ for all $v \in V$. We can form a graph G' from G by adding a node v and an edge uv of weight zero for every $u \in V$. It is easy to see that the minimum weight of a 3-cut of G' is the same as that of G . Also, if we map the new node to z , we get a triangle embedding of G' , and it corresponds to a feasible solution of (LP3) on G' having value equal to cx^* . Finally, every flat cut of G' corresponds to a flat cut of G of the same weight. So the theorem is

true for G if it is true for G' . It follows that we may assume that y^* is onto. Therefore, we may assume that $V = \Delta_q$, and that y^* is the identity mapping.

Now suppose that there exists $uv \in E \setminus E_q$, such that $c_{uv} = \varepsilon > 0$. Let P be the edge-set of a path in G_q from u to v such that $|P| = d_q(u, v)$. Decrease c_{uv} to zero, and increase c_e by ε for all $e \in P$. We denote the new c by c' . Then, since every 3-cut using e uses an edge from P , the minimum weight of a 3-cut with respect to c' is not less than that with respect to c . (Similarly, every flat 3-cut has value with respect to c' not less than that with respect to c .) Now $c'x^* = cx^* - \varepsilon d_q(u, v) + \varepsilon d_q(u, v) = cx^*$. This argument can be repeated as long as there is such an edge uv . □

It is a consequence of the above theorem that it is enough to study the 3-cut problem on graphs G_q with $x_e^* = \frac{1}{q}$ for all $e \in E_q$. (That is, to obtain the best bound on the ratio of the optimal weight of a 3-cut to the optimal value of (LP3), it suffices to analyze such graphs and weights.) In particular, for each positive integer q , we are interested in the optimal value of the following linear programming problem.

$$(P_q) \quad \begin{array}{ll} \text{minimize} & \frac{1}{q} \sum_{e \in E} c_e \\ \text{subject to} & \\ & c(S) \geq 1, \quad S \text{ a 3-cut of } G_q \\ & c_e \geq 0, \quad e \in E_q \end{array}$$

The dual problem is

$$(D_q) \quad \begin{array}{ll} \text{maximize} & \sum z_S \\ \text{subject to} & \\ & \sum_{e \in S} \leq \frac{1}{q}, \quad e \in E_q \\ & z_S \geq 0, \quad S \text{ a 3-cut of } G_q. \end{array}$$

We actually solved these problems numerically for several values of q , and then were able to find solutions for general q .

Theorem 3. *For $q \geq 4$ the optimal value of (P_q) and of (D_q) is equal to*

$$f(q) = \begin{cases} \frac{11}{12} + \frac{1}{12(q+1)}, & \text{if } q \equiv 0 \pmod{3} \\ \frac{11}{12} + \frac{1}{12q}, & \text{if } q \equiv 1 \pmod{3} \\ \frac{11}{12} + \frac{1}{12q} - \frac{1}{12q^2}, & \text{if } q \equiv 2 \pmod{3} \end{cases}$$

Moreover, there is an optimal solution of (D_q) for which z_S is positive only if S is a flat 3-cut.

It is easy to see that Theorems 2 and 3 have the following consequence. This result has been proved independently by Karger *et al.* [7], whose approach is somewhat different, but also uses a linear programming analysis of triangle-embedding.

Theorem 4. *For any 3-cut instance, the minimum weight of a 3-cut is at most $\frac{12}{11}$ times the optimal value of (LP3), and the constant $\frac{12}{11}$ is best possible. □*

4 An Improved Approximation Algorithm

Algorithm

1. Find a rational-valued optimal solution x^* of $(LP3)$.
2. Find the triangle embedding y^* determined by x^* .
3. Return the flat 3-cut of minimum weight.

As pointed out before, the first step can be performed in polynomial time. The polynomial-time algorithms for linear programming can be modified to return a rational-valued optimal solution, and one of polynomial size. The second is easy. So is the third step, using the observation made earlier that there are only $O(n^2)$ flat 3-cuts of G .

Theorem 5. *The above algorithm returns a 3-cut of weight at most $\frac{12}{11}$ times the minimum weight of a 3-cut.*

Proof. We may assume that the optimal value of a 3-cut is 1, so it is enough to prove that the algorithm delivers a 3-cut of weight at most $\frac{12}{11}$. Let x^* be a rational-valued optimal solution for $(LP3)$, and let q be a common denominator for the components of x^* , such that q is a multiple of 3. Consider an optimal solution z^* of (D_q) as given by Theorem 3. Then

$$\sum_S \frac{12}{11} z_S^* \geq 1,$$

and $z_S^* > 0$ only if S is a flat 3-cut. Therefore

$$\begin{aligned} \min_{z_S^* > 0} c(S) &\leq \sum_S \frac{12}{11} z_S^* c(S) \\ &= \frac{12}{11} \sum_S z_S^* c(S) \\ &= \frac{12}{11} \sum_{e \in E} c_e \sum_{e \in S} z_S^* \\ &\leq \frac{12}{11} \sum_{e \in E} c_e x_e^* \\ &\leq \frac{12}{11}. \end{aligned}$$

□

5 Proof of Theorem 3

To prove Theorem 3, it is enough to give feasible solutions of (P_q) and of (D_q) having objective value $f(q)$. For simplicity, we will actually do something weaker. For the case when $q \equiv 0 \pmod{3}$, we give a feasible solution of (P_q) having objective value $f(q)$, and a feasible solution to (D_q) using only variables corresponding

to flat 3-cuts having objective value $\frac{11}{12}$. Although this does not quite prove Theorem 3, it is enough to prove Theorems 4 and 5, since a common denominator for the components of x^* can always be chosen to be a multiple of 3.

First, we describe our feasible solution to (P_q) . Consider Figure 2 which shows G_9 . Let c'_e be the number next to edge e , or 1 if no number appears. It is easy to see that the minimum value of a 3-cut is 40, so $c = c'/40$ is a feasible solution to (P_9) . Its objective value is the sum of the components of c' divided by 9, which is $\frac{37}{40}$.

Here is the general construction (when q is a multiple of 3) for an optimal solution of (P_q) . If $q = 3m$, divide Δ_q into three “corner triangles” of side m together with the “middle hexagon”. Put $c'_e = 3m + 1$ for all edges incident with the terminals. Put $c'_e = 2m + 2$ for all other edges on the boundary of Δ_q . Put $c'_e = m - 1$ for each edge e in a corner triangle that is parallel to an outside edge and distance 1 from it. Put $c'_e = 1$ for all other edges in the middle hexagon (including its boundary). Put $c'_e = 0$ for all other edges.

It is easy to convince oneself that the minimum weight of a 3-cut with respect to c' is $4(3m + 1)$, and hence that $c = c'/4(3m + 1)$ is a feasible solution to (P_q) . Here is a sketch of a proof. (The ideas come, essentially, from the result of Dahlhaus, *et al.* [5], showing that there is a polynomial-time algorithm to solve the optimal multiterminal cut problem when G is planar and the number of terminals is fixed.) Any minimal 3-cut of G_q has the form $\beta(R_1, R_2, R_3)$. There are two kinds of such 3-cuts, corresponding to the case in which there is a pair i, j for which there is no edge joining a node in R_i to a node in R_j , and the one where this is not true. The minimum value of a 3-cut of the first type is simply the sum of the weights of two cuts, each separating a terminal from the other two. In the case of G_q with c' described above, to show that any such cut has weight at least $4(3m + 1)$, it is enough to show (due to the symmetry of c') that any cut separating one terminal from the other two has weight at least $2(3m + 1)$. This is done by exhibiting an appropriate flow of this value from one terminal to the other two.

The second type of 3-cut corresponds to the union of three paths in the planar dual of G_q , such that the three paths begin at the same face triangle and end with edges that are on different sides of the outside face. Finding a minimum-weight such 3-cut can be accomplished by, for each choice of the face triangle, solving a shortest path problem. Therefore, to show that any 3-cut of the second type has c' -weight at least $4(3m + 1)$, one shows that, for each choice of face triangle, there is an appropriate “potential” on the faces of G_q .

To compute the objective value of this feasible solution (P_q) , note that there are 6 edges e having $c'_e = 3m + 1$, $3(3m - 2)$ edges e having $c'_e = 2m + 2$, $6(m - 1)$ edges e having $c'_e = m - 1$, and $9m^2$ edges e having $c'_e = 1$. From this we get that the total c' -weight of all the edges is $3m(11m + 12)$. To obtain the objective value of the resulting c in (P_q) , we divide by $4(3m + 1)(3m)$, and this gives $f(q)$ for $q = 3m$.

Now we need to show a feasible solution of (D_q) having objective value $\frac{11}{12}$. This requires a weighting of the flat 3-cuts of G_q . We assign positive dual vari-

ables to two kinds of 3-cuts. For each integer j , $1 \leq j < m$ and each choice of two terminals r, s we consider the (uniform) 3-cut $\beta(R_r(j), R_s(j), V \setminus (R_r(j) \cup R_s(j)))$ where, for $t = r, s$, $R_t(j) = \{v \in V_q : d_q(t, v) < j\}$. There are $3m$ such 3-cuts S , and for each of them we set $z_S = \frac{1}{4q}$. Notice that these variables contribute to the left-hand side of the main constraint of (D_q) only for certain edges, namely, those that are contained in the corner triangles and are parallel to one of the two sides of $\triangle$ that meet at that corner. For each of these edges, the total contribution is exactly $1/2q$.

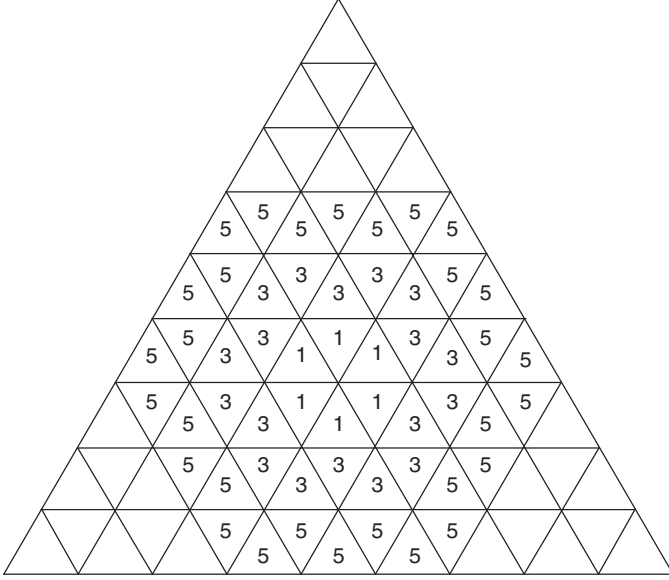


Fig. 3. Feasible solution of (D_9)

The weights assigned to the second type of flat cut are determined by a weighting of the face triangles of G_q that are contained in the middle hexagon. See Figure 3, where such a weighting of the face triangles is indicated for G_9 . Let us use the term *row* in the following technical sense. It is defined by a straight line through the centre of a face triangle and parallel to one of its three sides. When we speak of the face triangles *in* the row, we mean all of the face triangles that are intersected by the line. When we speak of the edges *in* the row, we mean all of the edges that are intersected by the line. Notice that in the figure, the sum of the weights of the face triangles in each row is the same, namely 35. It is obvious how to extend this pattern to find a weighting with this property for any $q = 3m$. Then the sum of the weights of the face triangles in any row is $4m^2 - 1$.

Given a face triangle, consider the set of all edges in the three rows containing the triangle. It is possible to choose two flat 3-cuts of G_q whose union is this

set, and whose intersection is a single edge, or is the set of edges of the face triangle. (There is more than one way to do this.) For each of these two 3-cuts, assign a weight equal to the weight of the triangle divided by $2q(4m^2 - 1)$. (Note that a 3-cut S may be assigned weight by two different face triangles; these weights are added to form the variable z_S .) Now consider the constraint of (D_q) corresponding to an edge e . The contribution of the variables just defined to the left-hand side of the constraint, is at most the sum of the weights of the face triangles in rows containing the edge. If the edge is in the middle hexagon, or is in a corner triangle and is not parallel to one of the edges incident with the corner, then it gets contributions from triangles in two different rows, and otherwise, it gets contributions from triangles in one row. Therefore, the contribution for the first type of edge is at most $(4m^2 - 1)/(4m^2 - 1)q = \frac{1}{q}$. For the second type of edge the total contribution is at most half this, that is, at most $\frac{1}{2q}$. But the second group of edges consists precisely of the ones that get a contribution from the dual variables assigned to the uniform 3-cuts, and that contribution is $\frac{1}{2q}$. So the total contribution of all of the dual variables to the left-hand side of the constraint of (D_q) corresponding to any edge e is at most $\frac{1}{q}$, so we have defined a feasible solution of (D_q) .

Now the objective value of this solution can be computed as follows. There are $3m$ variables corresponding to uniform 3-cuts, each given value $\frac{1}{4q}$. Therefore, the contribution to the objective function of variables of this type is $\frac{3m}{12m} = \frac{1}{4}$. The contribution of the other variables is the sum of over the $2m$ horizontal rows in the middle hexagon, of the total weight of a row divided by $q(4m^2 - 1)$. Therefore, it is

$$2m(4m^2 - 1)/q(4m^2 - 1) = \frac{2}{3}.$$

Therefore, the objective value of our feasible solution to (D_q) is

$$\frac{1}{4} + \frac{2}{3} = \frac{11}{12}.$$

6 Remarks

Since the constant $\frac{12}{11}$ is best possible in Theorem 4, it is natural to ask whether it is best possible in Theorem 5. Note, however, that the family of examples that we use to show the tightness of the LP bound, all have the property that there is a flat 3-cut that is optimal. Therefore, these examples are not at all bad for the approximation algorithm. However, it seems likely that $\frac{12}{11}$ is indeed best possible in Theorem 5. For several values of q Kevin Cheung [2] has constructed examples in which the optimal solution of $(LP3)$ has denominator q , and the approximation algorithm returns a 3-cut of value at least $\frac{1}{f(q)}$ times the optimal value of $(LP3)$. Actually, his examples seem to be the first that show that our approximation algorithm does not always return an optimal solution. In fact, no such example seems to have been known even for the simpler algorithm of Călinescu *et al.* [1].

All of the results of Călinescu *et al.* [1] quoted above for $k = 3$ are special cases of their results for general k . They give a linear-programming relaxation that generalizes (LP3), and a corresponding generalization of the notion of triangle-embedding, an embedding into a $(k - 1)$ -dimensional simplex in which the terminals are mapped to the extreme points. They show that the optimal value of a k -cut is at most $\frac{3k-2}{2k}$ times the optimal value of this linear-programming problem. As a result, they obtain an approximation algorithm for the optimal k -cut problem having performance guarantee $\frac{3k-2}{2k}$. The recent paper [7], which has most of our results for $k = 3$, also has results for $k > 3$, improving the bounds given by [1]. For example, [7] gives bounds of 1.1539 for $k = 4$ and 1.3438 for all $k > 6$. The problem of giving a tight analysis for $k > 3$, as we now have for $k = 3$, remains open.

Acknowledgment. We are grateful to Gruia Călinescu, Joseph Cheriyan, Kevin Cheung, and Levent Tunçel for conversations about this work.

References

1. G. Călinescu, H. Karloff, and Y. Rabani: An improved approximation algorithm for MULTIWAY CUT Proceedings of Symposium on Theory of Computing, ACM, 1998.
2. Kevin Cheung, private communication, 1999.
3. S. Chopra and M.R. Rao, "On the multiway cut polyhedron", Networks 21(1991), 51–89.
4. W.H. Cunningham, "The optimal multiterminal cut problem", in: C. Monma and F. Hwang (eds.), *Reliability of Computer and Communications Networks*, American Math. Soc., 1991, pp. 105–120.
5. E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis, "The Complexity of multiway cuts", extended abstract, 1983.
6. E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis, "The Complexity of multiterminal cuts", SIAM J. Computing, 23(1994), 864–894.
7. D. Karger, P. Klein, C. Stein, M. Thorup, and N. Young, "Rounding algorithms for a geometric embedding of minimum multiway cut," Proceedings of Symposium on Theory of Computing, ACM, 1999, to appear.

Semidefinite Programming Methods for the Symmetric Traveling Salesman Problem

Dragoš Cvetković¹, Mirjana Čangalović², and Vera Kovačević-Vujčić²

¹ Faculty of Electrical Engineering, University of Belgrade

² Faculty of Organizational Sciences, University of Belgrade

Abstract. In this paper the symmetric traveling salesman problem (STSP) is modeled as a problem of discrete semidefinite programming. A class of semidefinite relaxations of STSP model is defined and two variants of a branch-and-bound technique based on this class of relaxations are proposed. The results of preliminary numerical experiments with randomly generated problems are reported.

Keywords. Semidefinite programming, Traveling salesman problem, Branch-and-bound methods.

1 Introduction

Semidefinite programming (SDP) has many applications to various classes of optimization problems (see e.g. [33]). In particular, there is a growing interest in the application of SDP to combinatorial optimization, where it is used in order to get satisfactory bounds on the optimal objective function value (see [15], [31] for a survey). Some examples are recently introduced semidefinite relaxations for the max-cut problem (Goemans, Williamson [16]), graph colouring problem (Karger, Motwani, Sudan [20]) and traveling salesman problem (Cvetković, Čangalović, Kovačević-Vujčić [7], [8]). It is the purpose of this paper to investigate the power of semidefinite relaxations for traveling salesman problem in a branch-and-bound framework.

The traveling salesman problem (TSP) is one of the best-known NP-hard combinatorial optimization problems. There is an extensive literature on both theoretical and practical aspects of TSP. The most important theoretical results on TSP can be found in [24] (see also [4], [9]). A large number of both exact algorithms and heuristics for TSP have been proposed; for a review we refer to Laporte [22], [23]. We shall mention here only the most important approaches for finding an exact solution of the symmetric traveling salesman problem (STSP). Two classical relaxations of STSP have been extensively discussed in literature. The first exploits the fact that the cost of an optimal STSP-tour cannot be less than that of a shortest 1-tree. Several algorithms of branch-and-bound type are based on this relaxation first proposed by Christofides [3]. The basic algorithm was developed by Held and Karp [19] and further improved by Helbig-Hansen and Krarup [18], Smith and Thompson [32], Volgenant and Jonker [34], Gavish and Srikanth [14] and, more recently, Carpaneto, Fischetti and Toth [2].

The second relaxation is the linear programming relaxation of the 2- matching problem corresponding to STSP. This relaxation has been embedded in various optimization algorithms by gradually introducing violated subtour elimination constraints, integrality constraints, other types of valid inequalities, etc. Dantzig, Fulkerson and Johnson were the first to propose such an algorithm [11], which was followed by Martin [25], Miliotis [26], [27], Land [21], Crowder and Padberg [5], Padberg and Hong [28], Padberg and Rinaldi [29], [30], Grötschel and Holland [17], among the others. In this paper we propose a new class of branch-and-bound algorithms for STSP which is based on semidefinite relaxations .

The paper is organized as follows. In Section 2 we prove that STSP can be modeled as the problem of discrete semidefinite programming. The model is based on the notion of the Laplacian of graphs. A class of semidefinite relaxations of the discrete STSP model is presented and its relation to 1-tree and 2-matching relaxations is discussed. At the end of Section 2 a new type of cutting planes for the linear programming relaxation of STSP which are motivated by the structure of the semidefinite relaxation model is introduced. In Section 3 two variants of a branch-and-bound method based on the semidefinite relaxation are proposed and preliminary numerical results are reported.

2 A Class of Semidefinite Relaxations for STSP

Semidefinite relaxations developed here are based on the following results of Fiedler related to the Laplacian of graphs and algebraic connectivity (see [12], [13]).

Let $G = (V, E)$ be an undirected simple graph, where $V = \{1, \dots, n\}$ is the set of vertices and E is the set of edges. The Laplacian $L(G)$ of graph G is a symmetric matrix defined as $L(G) = D(G) - A(G)$, where $D(G)$ is the diagonal matrix with vertex degrees on the diagonal and $A(G)$ is the adjacency matrix of G .

The matrix $L(G)$ is positive semidefinite. If $\lambda_1 \leq \dots \leq \lambda_n$ are eigenvalues of $L(G)$ then $\lambda_1 = 0$ with the corresponding eigenvector $e = (1, \dots, 1)$. All other eigenvalues have eigenvectors which belong to set

$$S = \left\{ x = (x_1, \dots, x_n) \in \mathbb{R}^n \mid \sum_{i=1}^n x_i = 0, \sum_{i=1}^n x_i^2 = 1 \right\}$$

According to Fiedler, the second smallest eigenvalue λ_2 of $L(G)$, is called the algebraic connectivity of G and denoted by $a(G)$. In [13] the following results are proved:

Theorem 1. *The algebraic connectivity $a(G)$ has the properties:*

- (i) $a(G) = \min_{x \in S} x^T L(G) x$
- (ii) $a(G) \geq 0, a(G) > 0$ if and only if G is connected.

Fiedler shows that the notion of the Laplacian and the algebraic connectivity can be generalized to graphs with positively weighted edges.

A C -edge-weighted graph $G_C = (V, E, C)$ is defined by graph $G = (V, E)$ and a symmetric nonnegative weight matrix C such that $c_{ij} > 0$ if and only if $\{i, j\} \in E$. Now the Laplacian $L(G_C)$ is defined as $L(G_C) = \text{diag}(r_1, \dots, r_n) - C$, where r_i is the sum of the i -th row of C . The Laplacian $L(G_C)$ has similar characteristics as $L(G)$. Namely it is symmetric, positive semidefinite with the smallest eigenvalue $\lambda_1 = 0$ and the corresponding eigenvector e . As before, the algebraic connectivity $a(G_C)$ is the second smallest eigenvalue of $L(G_C)$, which enjoys similar properties to those in Theorem 1.

Theorem 2 ((M. Fiedler [13])). *The generalized algebraic connectivity $a(G_C)$ has the following properties:*

- (i) $a(G_C) = \min_{x \in S} x^T L(G_C) x$
- (ii) $a(G_C) \geq 0, a(G_C) > 0$ if and only if G_C is connected.

In the sequel we shall assume that $G = (V, E)$ is a complete undirected graph, where, as before, $V = \{1, \dots, n\}$ is the set of vertices and E is the set of edges. To each edge $\{i, j\} \in E$ a distance (cost) d_{ij} is associated such that the distance matrix $D = [d_{ij}]_{n \times n}$ is symmetric and $d_{ii} = 0, i = 1, \dots, n$. Now the symmetric traveling salesman problem (STSP) can be formulated as follows: *find a Hamiltonian circuit of G with minimal cost.*

Algebraic connectivity of a Hamiltonian circuit is well known in the theory of graph spectra (see e.g. [10]). The Laplacian of a circuit with n vertices has the spectrum

$$2 - 2\cos(2\pi j/n), \quad j = 1, \dots, n$$

and the second smallest eigenvalue is obtained for $j = 1$ and $j = n - 1$, i.e. $\lambda_2 = \lambda_3 = 2 - 2\cos(2\pi/n)$. This value will be denoted by h_n , i.e. $h_n = 2 - 2\cos(2\pi/n)$.

The next theorem, which gives a basis for the discrete semidefinite programming model of STSP, has been proved in [8] as a consequence of a more general result. For the sake of completeness we supply here a self-contained proof.

Theorem 3. *Let H be a spanning subgraph of G such that $d(i) = 2, i = 1, \dots, n$, where $d(i)$ is the degree of vertex i with respect to H , and let $L(H) = [l_{ij}]_{n \times n}$ be the corresponding Laplacian. Let α and β be real parameters such that $\alpha > h_n/n, 0 < \beta \leq h_n$. Then H is a Hamiltonian circuit if and only if the matrix $X = L(H) + \alpha J - \beta I$ is positive semidefinite, where J is the $n \times n$ matrix with all entries equal to one and I is the unit matrix of order n .*

Proof. Let $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ be the eigenvalues of $L(H)$ and let $x^1 = e$ and $x^i \in S, i = 2, \dots, n$, be the corresponding eigenvectors which form a basis for $\mathbb{R}^n$. It is easy to check that J has two eigenvalues: 0, with multiplicity $n - 1$ and the corresponding eigenvectors $x^2, \dots, x^n$, and n with e as its eigenvector. Therefore

$$\begin{aligned} Xe &= (L + \alpha J - \beta I)e = (\alpha n - \beta)e \\ Xx^i &= (L + \alpha J - \beta I)x^i = (\lambda_i - \beta)x^i, \quad i = 2, \dots, n \end{aligned}$$

which means that $\alpha n - \beta$ and $\lambda_i - \beta, i = 2, \dots, n$ are eigenvalues of X with eigenvectors $e, x^2, \dots, x^n$, respectively.

The conditions of Theorem 3 guarantee that H is a 2-matching, i.e. it is either a Hamiltonian circuit or a collection of at least two disjoint subcircuits. In the first case $\lambda_2 = h_n$, while in the second, according to Theorem 1, $\lambda_2 = 0$. As $\alpha > h_n/n$ in both cases it follows that $\alpha n - \beta > \lambda_2 - \beta$, i.e. the smallest eigenvalue of X is equal to $\lambda_2 - \beta$.

Suppose that H is a Hamiltonian circuit. Then $\beta \leq h_n$ implies $\lambda_2 - \beta = h_n - \beta \geq 0$, i.e. matrix X is positive semidefinite. Suppose now that X is positive semidefinite. Then $\lambda_2 - \beta \geq 0$ and $\beta > 0$ imply $\lambda_2 = a(H) > 0$ and by Theorem 1 it follows that H is a connected 2-matching, i.e. a Hamiltonian circuit. $\square$

It follows from Theorem 3 that a spanning subgraph H of G is a Hamiltonian circuit if and only if its Laplacian $L(H)$ satisfies the following conditions:

$$l_{ii} = 2, \quad i = 1, \dots, n \quad (1)$$

$$X = L(H) + \alpha J - \beta I \text{ is positive semidefinite, } \alpha > h_n/n, \quad 0 < \beta \leq h_n \quad (2)$$

Starting from (1) and (2) the following discrete semidefinite programming model of STSP can be defined

$$\text{minimize } F(X) = \sum_{i=1}^n \sum_{j=1}^n \left(-\frac{1}{2} d_{ij} \right) x_{ij} + \frac{\alpha}{2} \sum_{i=1}^n \sum_{j=1}^n d_{ij} \quad (3)$$

subject to

$$x_{ii} = 2 + \alpha - \beta, \quad i = 1, \dots, n \quad (4)$$

$$\sum_{j=1}^n x_{ij} = n\alpha - \beta, \quad i = 1, \dots, n \quad (5)$$

$$x_{ij} \in \{\alpha - 1, \alpha\}, \quad i, j = 1, \dots, n, \quad i < j \quad (6)$$

$$X \geq 0 \quad (7)$$

where $X \geq 0$ denotes that the matrix $X = [x_{ij}]_{n \times n}$ is symmetric and positive semidefinite and α and β are chosen according to Theorem 3. Matrix $L = X + \beta I - \alpha J$ represents the Laplacian of a Hamiltonian circuit if and only if X satisfies (4)-(7). Indeed, constraints (4)-(6) provide that L has the form of a Laplacian with diagonal entries equal to 2, while condition (7) guarantees that L corresponds to a Hamiltonian circuit. Therefore, if X^* is an optimal solution of problem (3)-(7) then $L^* = X^* + \beta I - \alpha J$ is the Laplacian of an optimal Hamiltonian circuit of G with the objective function value $\sum_{i=1}^n \sum_{j=1}^n \left(-\frac{1}{2} d_{ij} \right) l_{ij}^* = F(X^*)$.

A natural semidefinite relaxation of the traveling salesman problem is obtained when discrete conditions (6) are replaced by inequality conditions:

$$\text{minimize } F(X) \quad (8)$$

subject to

$$x_{ii} = 2 + \alpha - \beta, \quad i = 1, \dots, n \quad (9)$$

$$\sum_{j=1}^n x_{ij} = n\alpha - \beta, \quad i = 1, \dots, n \quad (10)$$

$$\alpha - 1 \leq x_{ij} \leq \alpha, \quad i, j = 1, \dots, n, i < j \quad (11)$$

$$X \geq 0 \quad (12)$$

It is easy to see that the relaxation (8)-(12) can be expressed in the standard form of an SDP problem. Indeed, constraint (9) can be represented as $A_i \circ X = 2 + \alpha - \beta$, where $\circ$ is the Frobenius product and A_i is a symmetric $n \times n$ matrix with 1 at the position (i, i) and all other entries equal to 0. Similarly, condition (10) is equivalent to $B_i \circ X = 2(n\alpha - \beta)$, where B_i has 2 at the position (i, i) while all the remaining elements of the i -th row and the i -th column are equal to 1, and all the other entries are zero. Finally, condition (11) can be expressed as $2(\alpha - 1) \leq C_{ij} \circ X \leq 2\alpha$, where C_{ij} has 1 at the positions (i, j) and (j, i) and zero otherwise. Since SDP problem (8)-(12) depends on parameters α and β it represents a class of semidefinite relaxations of TSP. In the sequel, members of this class will be referred to as SDP relaxations.

Let us denote by D and D° the feasible set of problem (8)-(12) and its relative interior. For each $X \in D$ the corresponding Laplacian $L = X + \beta I - \alpha J$ can be interpreted as the Laplacian of the weighted graph $G_L = (V, E_L, C_L)$, where $E_L = \{\{i, j\} \in E \mid l_{ij} < 0\}$ and $C_L = 2I - L$. If α and β satisfy the conditions of Theorem 3 then, using similar arguments as in the proof of Theorem 3, it can be shown that $X \geq 0$ is equivalent to $a(G_L) \geq \beta$ (see also [8]). Hence, by Theorem 2 graph G_L is connected. It immediately follows that 2-matchings with disjoint subcircuits cannot correspond to any X in D .

It is easy to see that $D^\circ \neq \emptyset$. Indeed, if e.g. $\hat{L} = \left(2 + \frac{2}{n-1}\right)I - \frac{2}{n-1}J$ then $\hat{X} = \hat{L} + \alpha J - \beta I = \left(2 + \frac{2}{n-1} - \beta\right)I + \left(\alpha - \frac{2}{n-1}\right)J$ has the eigenvalues $2 + \frac{2}{n-1} - \beta$ with the multiplicity $n - 1$ and $n\alpha - \beta$ with the multiplicity 1. Since $n\alpha - \beta > 0$ and $2 + \frac{2}{n-1} - \beta \geq 2 + \frac{2}{n-1} - h_n > 0$ for $n \geq 4$, it follows that $\hat{X} \in D^\circ$, $n \geq 4$.

For $\beta < h_n$ matrices X which correspond to Laplacians of Hamiltonian circuits are in D° , while for $\beta = h_n$ these matrices belong to $D \setminus D^\circ$. It is clear that the best relaxation is obtained for $\beta = h_n$. For that reason in numerical experiments reported in Section 3 parameter β is always chosen to be equal to h_n . Concerning the parameter α , it is always sufficient to choose $\alpha = 1$.

The semidefinite relaxation (8)-(12) is substantially different from the existing STSP relaxations. It should be pointed out that it cannot be theoretically compared neither with 2-matching nor with 1-tree. Indeed, if we consider STSP

model (3)-(7) it is easy to see that X which corresponds to the Laplacian of a 2-matching satisfies (4)-(6) but need not satisfy (7). In the case of 1-tree, the condition (4) is relaxed, while (5), (6) and (7) hold (see [8]). Preliminary numerical experiments on randomly generated problems with $10 \leq n \leq 20$ which are reported in [8], indicate that SDP relaxation gives considerably better lower bounds than both 1-tree and 2-matching.

SDP model (8)-(12) can be expressed in terms of Laplacians in the following equivalent way

$$\text{minimize } \Phi(L) = \sum_{i=1}^n \sum_{j=1}^n \left(-\frac{1}{2} d_{ij} \right) l_{ij} \quad (13)$$

subject to

$$l_{ii} = 2, \quad i = 1, \dots, n \quad (14)$$

$$\sum_{j=1}^n l_{ij} = 0, \quad i = 1, \dots, n \quad (15)$$

$$-1 \leq l_{ij} \leq 0, \quad i, j = 1, \dots, n, i < j \quad (16)$$

$$\lambda_2(L) \geq \beta \quad (17)$$

where $L = [l_{ij}]_{n \times n}$ is a symmetric matrix with second smallest eigenvalue $\lambda_2(L)$. The existing branch-and-cut approaches to STSP start from the linear relaxation of the 2-matching problem defined by (13) - (16) and form a search tree branching on fractional variables. At each node of the tree several types of cutting planes (subtour elimination constraints, 2-matching inequalities, comb inequalities, etc.) are introduced in order to limit the growth of the search tree. None of these explicitly takes in account nonlinear constraint (17). In the sequel we shall discuss a possibility to introduce a new type of cutting planes based on (17). Suppose that in (17) β is chosen to be equal to h_n and let us denote by V and W the feasible sets of problems (13) - (16) and (13) - (17), respectively. It is clear that $V \setminus W \neq \emptyset$. The following theorem can be proved.

Theorem 4. *Let $L^* \in V \setminus W$ and let $s \in S$ be an eigenvector corresponding to $\lambda_2(L^*)$. Then*

$$(i) \quad L \circ ss^T \geq h_n \text{ for each } L \in W$$

$$(ii) \quad \text{The hyperplane } L \circ ss^T = h_n \text{ is supporting for the set } W.$$

Proof. (i) For each $L \in W$, according to Theorem 2, we have

$$\lambda_2(L) = \min_{x \in S} x^T L x = \min_{x \in S} L \circ x x^T \geq h_n.$$

As $s \in S$ then $L \circ ss^T \geq h_n$.

(ii) It is sufficient to prove that the hyperplane $L \circ ss^T = h_n$ and the set W have nonempty intersection. We shall construct a point in the intersection of the form $L_\gamma = \gamma L^* + (1 - \gamma) \hat{L}$, where $\hat{L} = \left(2 + \frac{2}{n-1} \right) I - \frac{2}{n-1} J$. The matrix

$\hat{L}$ has the eigenvalue $2 + \frac{2}{n-1}$ with multiplicity $n-1$ and each $x \in S$ as its eigenvector, and 0 with multiplicity 1. Since $\lambda_2(\hat{L}) = 2 + \frac{2}{n-1} > h_n$ it follows that $\hat{L} \in \text{ri } W$.

Let us prove that for each $\gamma \in (0, 1)$ the vector s is an eigenvector corresponding to $\lambda_2(L_\gamma)$. Indeed, for each $x \in S$

$$L_\gamma \circ xx^T = \gamma L^* \circ xx^T + (1-\gamma)\hat{L} \circ xx^T \geq \gamma L^* \circ ss^T + (1-\gamma)\hat{L} \circ ss^T = L_\gamma \circ ss^T.$$

Moreover,

$$\lambda_2(L_\gamma) = L_\gamma \circ ss^T = \gamma \lambda_2(L^*) + (1-\gamma) \left(2 + \frac{2}{n-1} \right).$$

For $\bar{\gamma} = \left(2 + \frac{2}{n-1} - h_n \right) / \left(2 + \frac{2}{n-1} - \lambda_2(L^*) \right)$ we have $\lambda_2(L_{\bar{\gamma}}) = L_{\bar{\gamma}} \circ ss^T = h_n$. $\square$

Suppose that L^* is an optimal solution obtained at a node of the search tree in some of existing branch-and-cut procedures. If $\lambda_2(L^*) < h_n$ and $s \in S$ is a corresponding eigenvector then

$$L \circ ss^T \geq h_n \tag{18}$$

is a cutting plane inequality which could be added to the current relaxation problem. Indeed, $\lambda_2(L^*) = L^* \circ ss^T < h_n$ and, according to (i) of Theorem 4, $L(H) \circ ss^T \geq h_n$ for each $L(H)$ which represents the Laplacian of a Hamiltonian circuit H with n vertices. Let us note that semidefinite relaxation (13)-(17) includes through (17) all valid inequalities of the type (18), which, according to (ii) of Theorem 4, are defined by supporting hyperplanes of the feasible set.

3 Branch and Bound Algorithms

The goal of our numerical experiments was to give a first insight in the power of SDP as a relaxation in branch and bound framework for STSP.

We have implemented two branch and bound algorithms with the SDP relaxation (with $\alpha = 1$, $\beta = h_n$) and one with the 1-tree relaxation. The last one was implemented to check the correctness of the results. All algorithms are based on the general branch and bound scheme as described in [24]. We used a FORTRAN implementation of the branch and bound shell from the package TSP-SOLVER [6], [9]. An initial upper bound was obtained in all cases by the 3-optimal heuristic. The depth first search was used to select the next subproblem.

The two branch and bound algorithms differ only in their branching rules:

Algorithm 1. At the first vertex of degree greater than 2 in the weighted graph representing the SDP solution an edge is excluded in each son;

Algorithm 2. The first non-integer entry of the SDP solution matrix is replaced in the sons by 0 and 1 respectively.

For solving the SDP relaxation tasks we used CSDP 2.2 software package developed by Borchers [1] in C language. Inequality conditions (11) were handled adding $n^2 - n$ slack variables each represented by a 1×1 block as accepted by the software.

Our numerical experiments included 55 randomly generated STSP instances of dimension $10 \leq n \leq 20$ already treated in [8]. Entries of the distance matrix are uniformly distributed in the range from 1 to 999. The experiments were performed on an Alpha 800 5/400 computer. In a time sharing system it took no more than 1 minute real time to get a solution of an SDP relaxation task related to an STSP instance of dimension $n \leq 20$. Computational results are presented in Table 1.

Table 1.

1	2	3	4	5	6	7	8	9
10	1	1680.9950	1681	1681	1	/	1	/
	2	2777.9920	2778	2778	1	/	1	/
	3	1626.1300	1714	1630	7	5	5	3
	4	2058.9950	2059	2059	1	/	1	/
	5	2672.4910	2801	2713	24	18	11(1)	6
11	1	2884.0000	2884	2884	1	/	1	/
	2	2258.4940	2283	2283	10	7	7	4
	3	1565.0000	1565	1565	1	/	1	/
	4	1226.8920	1229	1229	7	5	5	3
	5	1999.0000	2019	2019	4	3	3	2
12	1	2962.0000	2962	2962	1	/	1	/
	2	2416.0000	2416	2416	1	/	1	/
	3	1267.0010	1267	1267	1	/	1	/
	4	2434.0000	2434	2434	1	/	1	/
	5	1981.7260	2021	2021	15	11	5	3
13	1	1742.0000	1742	1742	1	/	1	/
	2	2064.4350	2072	2072	49	35	7(2)	4
	3	1786.0010	1786	1786	1	/	1	/
	4	2650.3250	2688	2686	10	7	11(1)	6
	5	2458.0000	2458	2458	1	/	1	/
14	1	1503.0000	1838	1503	1	/	1	/
	2	2269.0000	2269	2269	1	/	1	/
	3	1985.5090	2091	2091	351	242	57(51)	29
	4	2170.5680	2173	2173	11	8	5	3
	5	2000.0000	2000	2000	1	/	1	/
15	1	1548.0010	1926	1548	1	/	1	/
	2	1415.0000	1415	1415	1	/	1	/
	3	1813.0000	2082	1849	4	3	3	2
	4	2455.6730	2471	2471	8	6	7(2)	4
	5	1749.0000	1749	1749	1	/	1	/

Table 1. (continued)

1	2	3	4	5	6	7	8	9
16	1	2579.0000	2579	2579	1	/	1	/
	2	2189.0000	2189	2189	1	/	1	/
	3	2147.9210	2247	2181	54	38	51(20)	26
	4	1447.7110	1473	1473	11	8	7(1)	4
	5	2595.0010	2896	2595	1	/	1	/
17	1	1183.9970	1184	1184	1	/	1	/
	2	2606.9930	2607	2607	1	/	1	/
	3	1664.9950	1665	1665	1	/	1	/
	4	1568.4940	1579	1576	13	9	9	5
	5	2192.2420	2233	2208	53	36	5(1)*	3
18	1	2606.5870	2676	2651	*		59(1)	30
	2	2273.2220	2356	2275	21	15	11	6
	3	1562.0080	1673	1562	1	/	1	/
	4	2490.0000	2490	2490	1	/	1	/
	5	1815.9110	1994	1824	*		17	9
19	1	1223.9960	1420	1224	1	/	1	/
	2	2039.9930	2073	2068	89	60	19(3)*	10
	3	1417.9960	1418	1418	1	/	1	/
	4	1897.4480	2121	1926	*		29(5)	15
	5	2015.5880	2166	2035	*		15(5)	8
20	1	1953.2990	2250	2011	*		71(19)	36
	2	2410.2790	2501	2420	452	309	7(3)	4
	3	2585.1740	2680	2589	4	3	13(2)	7
	4	1758.4360	1784	1777	20	14	15(2)	8
	5	1817.7610	1838	1838	50	34	27(1)	14

The columns in Table 1 contain the following data:

1. Dimension of TSP (the number of cities);
2. Instance identification number;
3. Value of the SDP relaxation (lower bound);
4. Value obtained by the 3-optimal heuristic (upper bound);
5. Length of the optimal solution;
6. The number of the solved relaxation tasks in Algorithm 1; asterisk indicates that this number is greater than 1000;
7. The number of killed subproblems in the Algorithm 1;
8. The number of solved relaxation tasks in Algorithm 2;
9. The number of killed subproblems in Algorithm 2.

It can be seen from Table 1 that Algorithm 2 is superior to Algorithm 1 with respect to the cardinality of the search tree. However, in the case of Algorithm 2 certain numerical instabilities occurred in solving some SDP relaxation tasks (CSDP output code 6). Numbers in parentheses in column 8 indicates the number of such tasks. In all of these cases the solutions of SDP problems were reached, but instead of usual 6-8 correct significant digits, only 3-5 correct digits were obtained. Insufficient accuracy influenced the search procedure only for 2

instances, denoted by asterisk in column 8, when Algorithm 2 failed to generate an optimal solution.

The software we have implemented for the experiments presented in this paper is far from being optimized. Improvements could include more sophisticated branching rules and additional heuristics for choosing the next subproblem, as well as a special purpose code adapted to the structure of SDP relaxations with stabilization mechanisms for handling numerical instabilities.

References

1. Borchers, B.: CSDP, A C Library for Semidefinite Programming. Optimization Methods and Software (to appear)
2. Carpaneto G., Fischetti M., Toth P.: New Lower Bounds for the Symmetric Travelling Salesman Problem. *Math. Program.* **45** (1989) 233–254.
3. Christofides N.: The Shortest Hamiltonian Chain of a Graph. *SIAM J. Appl. Math.* **19** (1970) 689–696.
4. Cook, W., Cunningham, W., Pulleyblank, W., Schrijver, A.: Combinatorial Optimization. John Wiley & Sons, New York Chichester Weinheim Brisbane Singapore Toronto (1998)
5. Crowder H., Padberg M.W.: Solving Large-Scale Symmetric Travelling Salesman Problems to Optimality. *Management Sci.* **26** (1980) 495–509
6. Cvetković, D., Čangalović, M., Dimitrijević, V., Kraus, L., Milosavljević, M., Simić, S.: TSP-SOLVER - A Programming Package for the Traveling Salesman Problem. Univ. Beograd, Publ. Elektrotehn. Fak. Ser. Mat., **1** (1990) 41–47
7. Cvetković, D., Čangalović, M., Kovačević-Vujčić, V.: Semidefinite Programming and Traveling Salesman Problem. In: Petrović, R., Radojević, D. (eds.): Proceedings of Yugoslav Symposium on Operations Research. Herceg Novi, Yugoslavia (1998) 239–242
8. Cvetković, D., Čangalović, M., Kovačević-Vujčić, V.: Semidefinite Relaxations of Travelling Salesman Problem. (to appear)
9. Cvetković, D., Dimitrijević, V., Milosavljević, M.: Variations on the Travelling Salesman Theme. *Libra Produkt*, Belgrade (1996)
10. Cvetković, D., Doob, M., Sachs, H.: Spectra of Graphs. 3rd edn. Johann Ambrosius Barth, Heidelberg Leipzig (1995)
11. Dantzig G.B., Fulkerson D.R., Johnson S.M.: Solution of a Large-Scale Traveling Salesman Problem. *Operations Research* **2** (1954) 393–410
12. Fiedler M.: Algebraic Connectivity of Graphs. *Czechoslovak Math. J.* **23** (1973) 298–305
13. Fiedler, M.: Laplacian of Graphs and Algebraic Connectivity. In: Combinatorics and Graph Theory, Vol. 25, Banach center publications, PWN-Polish scientific publishers Warsaw (1989) 57–70
14. Gavish B., Srikant K.N.: An Optimal Solution Method for Large-Scale Multiple Travelling Salesman Problems. *Operations Research* **34** (1986) 698–717
15. Goemans, M.: Semidefinite Programming in Combinatorial Optimization. *Math. Program.* **79** (1997) 143–161
16. Goemans M.X., Williamson D.P.: Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM* **42** (1995) 1115–1145

17. Grötschel M., Holland O.: Solution of Large-Scale Symmetric Travelling Salesman Problems. *Math. Program.* **51** (1991) 141–202
18. Helbig-Hansen K., Krarup J.: Improvements of the Held-Karp Algorithm for the Symmetric Traveling Salesman Problem. *Math. Program.* **7** (1974) 87–96
19. Held M., Karp R.M.: The Travelling Salesman Problem and Minimum Spanning Trees. Part II, *Math. Program.* **1** (1971) 6–25
20. Karger D., Motwani R., Sudan M.: Approximate Graph Coloring by Semidefinite Programming. *J. ACM* **45** (1998) 246–265
21. Land A.H.: The Solution of Some 100-City Travelling Salesman Problems. Working Paper. London School of Economics (1979)
22. Laporte, G.: The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms. *European J. Operational Research* **59** (1992) 231–247
23. Laporte G.: Exact Algorithms for the Traveling Salesman Problem and the Vehicle Routing Problem. *Les Cahiers du GERAD G-98-37* July (1998)
24. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: The Traveling Salesman Problem. John Wiley & Sons, Chichester New York Brisbane Toronto Singapore (1985)
25. Martin G.T.: Solving the Travelling Salesman Problem by Integer Programming. Working Paper. CEIR, New York (1966)
26. Miliotis P.: Integer Programming Approaches to the Travelling Salesman Problem. *Math. Program.* **10** (1976) 367–378
27. Miliotis P.: Using Cutting Planes to Solve the Symmetric Travelling Salesman Problem. *Math. Program.* **15** (1978) 177–188
28. Padberg M.W., Hong S.: On the Symmetric Travelling Salesman Problem: A Computational Study. *Math. Program. Study* **12** (1980) 78–107
29. Padberg M.W., Rinaldi G.: Optimization of a 532-City Symmetric Traveling Salesman Problem by Branch and Cut. *Operations Research Letters* **6** (1987) 1–7
30. Padberg M.W., Rinaldi G.: A Branch-and-Cut Algorithm for the Resolution of Large Scale Symmetric Traveling Salesman Problems. *SIAM Review* **33** (1991) 66–100
31. Rendl, F.: Semidefinite Programming and Combinatorial Optimization. Technical Report Woe-19, TU Graz, Austria December (1997)
32. Smith T.H.C., Thompson G.L.: A LIFO Implicit Enumeration Search Algorithm for the Symmetric Traveling Salesman Problem Using Held and Karp's 1-Tree Relaxation. *Annals Disc. Math.* **1** (1977) 479–493
33. Vandenberghe, L., Boyd, S.: Semidefinite Programming. *SIAM Review* **38** (1996) 49–95
34. Volgenant T., Jonker R.: A Branch and Bound Algorithm for the Symmetric Traveling Salesman Problem Based on the 1-Tree Relaxation. *European J. Operational Research* **9** (1982) 83–89

Bounds on the Chvátal Rank of Polytopes in the 0/1-Cube

Friedrich Eisenbrand¹ and Andreas S. Schulz²

¹ Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany,
`eisen@mpi-sb.mpg.de`

² MIT, Sloan School of Management and Operations Research Center, E53-361,
Cambridge, MA 02139, USA, `schulz@mit.edu`

Abstract. Gomory's and Chvátal's cutting-plane procedure proves recursively the validity of linear inequalities for the integer hull of a given polyhedron. The number of rounds needed to obtain all valid inequalities is known as the Chvátal rank of the polyhedron. It is well-known that the Chvátal rank can be arbitrarily large, even if the polyhedron is bounded, if it is of dimension 2, and if its integer hull is a 0/1-polytope. We prove that the Chvátal rank of polyhedra featured in common relaxations of many combinatorial optimization problems is rather small; in fact, the rank of any polytope contained in the n -dimensional 0/1-cube is at most $3n^2 \lg n$. This improves upon a recent result of Bockmayr et al. [6] who obtained an upper bound of $O(n^3 \lg n)$.

Moreover, we refine this result by showing that the rank of any polytope in the 0/1-cube that is defined by inequalities with small coefficients is $O(n)$. The latter observation explains why for most cutting planes derived in polyhedral studies of several popular combinatorial optimization problems only linear growth has been observed (see, e.g., [13]); the coefficients of the corresponding inequalities are usually small. Similar results were only known for monotone polyhedra before.

Finally, we provide a family of polytopes contained in the 0/1-cube the Chvátal rank of which is at least $(1 + \epsilon)n$ for some $\epsilon > 0$; the best known lower bound was n .

1 Introduction

Chvátal [11] established cutting-plane proofs as a way to certify certain properties of combinatorial problems, e.g., to testify that there are no k pairwise non-adjacent nodes in a given graph, that there is no acyclic subdigraph with k arcs in a given digraph, or that there is no tour of length at most k in a prescribed instance of the traveling salesperson problem. In this paper we discuss the length of such proofs. Let us first recall the notion of a cutting-plane proof. A sequence of inequalities

$$c_1 x \leq \delta_1, c_2 x \leq \delta_2, \dots, c_m x \leq \delta_m \tag{1}$$

is called a *cutting-plane proof* of $c x \leq \delta$ from a given system of linear inequalities $Ax \leq b$, if $c_1, \dots, c_m$ are integral, $c_m = c$, $\delta_m = \delta$, and if $c_i x \leq \delta'_i$ is a

nonnegative linear combination of $Ax \leq b$, $c_1 x \leq \delta_1, \dots, c_{i-1} x \leq \delta_{i-1}$ for some δ'_i with $\lfloor \delta'_i \rfloor \leq \delta_i$. Obviously, if there is a cutting-plane proof of $c x \leq \delta$ from $Ax \leq b$ then every integer solution to $Ax \leq b$ must satisfy $c x \leq \delta$. Chvátal [11] showed that the converse holds as well. That is, if all integer points in a nonempty polytope $\{x \in \mathbb{R}^n : Ax \leq b\}$ satisfy an inequality $c x \leq \delta$, for some $c \in \mathbb{Z}^n$, then there is a cutting-plane proof of $c x \leq \delta$ from $Ax \leq b$. Schrijver extended this result to rational polyhedra [36].

In a way, the sequential order of the inequalities in (1) obscures the (recursive) structure of the cutting-plane proof; it is better revealed by a directed graph with vertices $0, 1, 2, \dots, m$, in which an arc goes from node i to node j iff the i -th inequality has a positive coefficient in the linear combination of the j -th inequality. Here, 0 serves as a representative for any inequality in $Ax \leq b$. The number of arcs in a longest simple path terminating at a node i is usually referred to as the *depth* of the i -th inequality $c_i x \leq \delta_i$ w.r.t. the cutting plane proof. The depth of the m -th inequality is called the *depth of the proof*, whereas m is the so-called *length* of the cutting-plane proof. We also say that an inequality $c x \leq \delta$ has depth (at most) d relative to a polyhedron $\{x : Ax \leq b\}$ if it has a cutting-plane proof from $Ax \leq b$ of depth less than or equal to d . The following theorem clarifies the relation between the depth and the length of a cutting-plane proof. It resembles very much the relation between the height and the number of nodes of a recursion tree where every interior node has at most degree n . It can be proved with the help of Farkas' Lemma.

Theorem 1 (Chvátal, Cook, and Hartmann [13]). *Let $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$, let $Ax \leq b$ have an integer solution, and let $c x \leq \delta$ have depth at most d relative to $Ax \leq b$. Then there is a cutting-plane proof of $c x \leq \delta$ from $Ax \leq b$ of length at most $(n^{d+1} - 1)/(n - 1)$.*

Gomory-Chvátal cutting-planes have gained importance for at least three reasons. First, the cutting-plane method is a (theoretical) tool to obtain a linear description of the integer hull of a polyhedron. In fact, as we already mentioned before any valid inequality for the integer hull has a cutting-plane proof from the defining system of the polyhedron. The *Chvátal rank* of this polyhedron is the smallest number d such that all inequalities valid for its integer hull have depth at most d relative to the defining system. Hence, if we later state lower and upper bounds for the depth of inequalities they immediately apply to the Chvátal rank of the corresponding polyhedron as well. Second, despite the early disappointments with Gomory's cutting-plane method [21, 22], it is of practical relevance. On the one hand, it has stimulated to a certain extent the search for problem-specific cutting planes which became the basis of an own branch of combinatorial optimization, namely polyhedral combinatorics (see, e.g., [33, 23, 35]). On the other hand, Balas et al. [2] successfully incorporated Gomory's mixed integer cuts within a Branch-and-Cut framework. Third, since cutting-plane theory implies that certain implications in integer linear programming have cutting-plane proofs, it is of particular importance in mathematical logic and complexity theory. It is a fundamental problem whether there exists a proof

system for propositional logic in which every tautology has a short proof. Here, the length of the proof is measured by the total number of symbols in it and short means polynomial in the length of the tautology. This question is equivalent to whether or not NP equals co-NP. Cook, Coullard, and Turán [14] were the first to consider cutting-plane proofs as a propositional proof system. In particular, they pointed out that the cutting-plane proof system is a strengthening of resolution proofs. Since the work of Haken [25] exponential lower bounds are known for the latter. Results of Chvátal, Cook, and Hartmann [13], of Bonet, Pitassi, and Raz [7], of Impagliazzo, Pitassi, and Urquhart [30], and of Pudlák [34] imply exponential lower bounds on the length of cutting-plane proofs as well. On the other hand, there is no upper bound on the length of cutting-plane proofs in terms of the dimension of the corresponding polyhedron as the following well-known example shows. The Chvátal rank of the polytope defined by

$$\begin{aligned} -t x_1 + x_2 &\leq 1 \\ t x_1 + x_2 &\leq t + 1 \\ x_1 &\leq 1 \\ x_1, x_2 &\geq 0 \end{aligned}$$

grows with t . Here, t is an arbitrary positive number. This fact is rather counter-intuitive since the corresponding integer hull is a 0/1-polytope, i.e., all its vertices have components 0 or 1 only. That is, for any 0/1-polytope there is a simple certificate of the validity of an inequality $cx \leq \delta$. Just list all, at most 2^n possible assignments of 0/1-values to the variables. One of our main results helps to meet the natural expectation. We give a polynomial bound in the dimension for the Chvátal rank of any polytope contained in the 0/1-cube. Then, Theorem 1 implies the existence of exponentially long cutting-plane proofs, matching the known exponential lower bounds.

In polyhedral combinatorics, it has been quite common to consider the depth of a class of inequalities if not as an indicator of quality at least as a measure of its complexity. Hartmann, Queyranne, and Wang [29] give conditions under which an inequality has depth at most 1 and use them to establish that several classes of inequalities for the traveling salesperson polytopes have depth at least 2, as was claimed before in [3, 8, 9, 10, 18, 20, 24]. However, it follows from a recent result in [16] that deciding whether a given inequality $cx \leq \delta$ has depth at least 2 can in general not be done in polynomial time, unless $P = NP$. Chvátal, Cook, and Hartmann [13] (see also [27]) answered questions and proved conjectures of Schrijver, of Barahona, Grötschel, and Mahjoub [4], of Jünger, of Chvátal [12], and of Grötschel and Pulleyblank [24] on the behavior of the depth of certain inequalities relative to popular relaxations of the stable set polytope, the bipartite-subgraph polytope, the acyclic-subdigraph polytope, and the traveling salesperson polytope, resp. They obtained similar results for the set-covering and the set-partitioning polytope, the knapsack polytope, and the maximum-cut polytope, and so did Schulz [38] for the transitive packing, the clique partitioning, and the interval order polytope. The observed increase of the depth was never faster than a linear function of the dimension; we prove

that this indeed has to be the case as the depth of any inequality with coefficients bounded by a constant is $O(n)$, relative to a polytope in the 0/1-cube. Naturally, most polytopes associated with combinatorial optimization problems are 0/1-polytopes.

Main Results. We present two new upper bounds on the depth of inequalities relative to polytopes in the 0/1-cube. For notational convenience, let P be any polytope contained in the 0/1-cube, i.e., $P \subseteq [0, 1]^n$, and let $cx \leq \delta$, $c \in \mathbb{Z}^n$ be an arbitrary inequality valid for the integer hull P_I of P .

We prove first that the depth of $cx \leq \delta$ relative to P is at most $2(n^2 + n \lg \|c\|_\infty)$. This yields an $O(n^2 \lg n)$ bound on the Chvátal rank of P since any 0/1-polytope P_I can be represented by a system of inequalities $Ax \leq b$ with $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ such that each absolute value of an entry in A is bounded by $n^{n/2}$. Note that the latter bound is sharp, i.e., there exist 0/1-polytopes with facets for which any inducing inequality $ax \leq \beta$, $a \in \mathbb{Z}^n$ satisfies $\|a\|_\infty \in \Omega(n^{n/2})$ [1].

Second, we show that the depth of $cx \leq \delta$ relative to P is no more than $\|c\|_1 + n$. A similar result was only known for monotone polyhedra [13]. In fact, we present a reduction to the monotone case that is of interest in its own right because of the smooth interplay of unimodular transformations and rounding operations. The second bound gives an asymptotic improvement by a factor n to the before-mentioned bound if the components of c are bounded by a constant.

Third, we construct a family of polytopes in the n -dimensional 0/1-cube whose Chvátal rank is at least $(1 + \epsilon)n$, for some $\epsilon > 0$. In other words, if $r(n)$ denotes the maximum Chvátal rank over all polytopes that are contained in $[0, 1]^n$, then it is one outcome of our study that this function behaves as follows:

$$(1 + \epsilon)n \leq r(n) \leq 3n^2 \lg n .$$

Finally, we also show that the number of inequalities in any linear description of a polytope $P \subseteq [0, 1]^n$ with empty integer hull is exponential in n , whenever there is an inequality of depth n .

Related Work. Via a geometric argument, Bockmayr and Eisenbrand [5] derived the first polynomial upper bound of $6n^3 \lg n$ on the Chvátal rank of polytopes in the n -dimensional 0/1-cube. Subsequently, Schulz [39] and Hartmann [28] independently obtained both a considerably simpler proof and a slightly better bound of $n^2 \lg(n^{n/2})$, by using bit-scaling. The reader is referred to the joint journal version of their papers [6], where the authors actually show that the depth of any inequality $cx \leq \delta$, $c \in \mathbb{Z}^n$, which is valid for P_I is at most $n^2 \lg \|c\|_\infty$, relative to P . For monotone polytopes P , Chvátal, Cook, and Hartmann [13] showed that the depth of any inequality $cx \leq \delta$ that is valid for P_I is at most $\|c\|_1$. Moreover, they also identified polytopes stemming from relaxations of combinatorial optimization problems that have Chvátal rank at least n .

Eventually, our study of $r(n)$ can also be seen as a continuation of the investigation of combinatorial properties of 0/1-polytopes, like their diameter [32],

their number of facets [19], their number of vertices in a 2-dimensional projection [31], or their feature of admitting polynomial-time simplex-type algorithms for optimization [40].

The paper is organized as follows. We start with some preliminaries and introduce some notation in Section 2. We also show that any linear description of a polytope in the 0/1-cube that has empty integer hull and Chvátal rank n needs to contain at least 2^n inequalities. In Section 3, we prove the $O(n^2 \lg n)$ upper bound on the Chvátal rank of polytopes in the 0/1-cube. Then, in Section 4, we utilize unimodular transformations as a key tool to derive an $O(n)$ bound on the depth of inequalities with small coefficients, relative to polytopes in the 0/1-cube. Finally, we present the new lower bound on the Chvátal rank in Section 5.

2 Preliminaries

A *polyhedron* P is a set of points of the form $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, for some matrix $A \in \mathbb{R}^{m \times n}$ and some vector $b \in \mathbb{R}^m$. The polyhedron is *rational* if both A and b can be chosen to be rational. If P is bounded, then P is called a *polytope*. The *integer hull* P_I of a polyhedron P is the convex hull of the integer points in P .

The *half space* $H = (cx \leq \delta)$ is the set $\{x \in \mathbb{R}^n \mid cx \leq \delta\}$, for some non-zero vector $c \in \mathbb{Q}^n$. It is called *valid* for a subset S of $\mathbb{R}^n$, if $S \subseteq H$. Sometimes we also say that the inequality $cx \leq \delta$ is valid for S . If the components of c are relatively prime integers, i.e., $c \in \mathbb{Z}^n$ and $\gcd(c) = 1$, then $H_I = (cx \leq \lfloor \delta \rfloor)$, where $\lfloor \delta \rfloor$ is the largest integer number less than or equal to δ . The *elementary closure* of a polyhedron P is the set

$$P' = \bigcap_{H \supseteq P} H_I,$$

where the intersection ranges over all rational half spaces containing P . We refer to an application of the $'$ operation as one iteration of the Gomory-Chvátal procedure. If we set $P^{(0)} = P$ and $P^{(i+1)} = (P^{(i)})'$, for $i \geq 0$, then the Chvátal rank of P is the smallest number t such that $P^{(t)} = P_I$. The depth of an inequality $cx \leq \delta$ with respect to P is the smallest k such that $cx \leq \delta$ is valid for $P^{(k)}$.

Let $P \subseteq \mathbb{R}^n$ be a polyhedron. A polyhedron Q with $Q \supseteq P$ is called a *weakening* of P , if $Q_I = P_I$. If $cx \leq \delta$ is valid for P_I , then the depth of this inequality with respect to Q is an upper bound on the depth of this inequality with respect to P . It is easy to see that each polytope $P \subseteq [0, 1]^n$ has a rational weakening in the 0/1-cube.

The following important lemma can be found in [37, p. 340]. (For a very nice treatment, see also [15, Lemma 6.33].) It allows to use induction on the dimension of the polyhedra considered and provides the key for the termination of the Gomory-Chvátal procedure, which was shown by Schrijver for rational polyhedra in [36].

Lemma 1. *Let F be a face of a rational polyhedron P . Then $F' = P' \cap F$.*

Lemma 1 yields the following upper bound on the Chvátal rank of rational polytopes in the 0/1-cube with empty integer hull (see [6] for details).

Lemma 2. *Let $P \subseteq [0, 1]^n$ be a d -dimensional rational polytope in the 0/1-cube with $P_I = \emptyset$. If $d = 0$, then $P' = \emptyset$; if $d > 0$, then $P^{(d)} = \emptyset$.*

Thus, if $cx \leq \delta$ is valid for a rational polytope $P \subseteq [0, 1]^n$ and $cx \leq \delta - 1$ is valid for P_I , then $cx \leq \delta - 1$ is valid for $P^{(n)}$.

With these methods at hand one can prove the following result due to Hartmann [27].

Lemma 3. *If $P \subseteq [0, 1]^n$ is a polytope and $\sum_{i \in I} x_i - \sum_{j \in J} x_j \leq r$ is valid for P_I for some subsets I and J of $\{1, \dots, n\}$, then this inequality has depth at most n^2 with respect to P .*

A side-product of our result in Section 4.3 is a reduction of this bound to $2n$.

Chvátal, Cook, and Hartmann [13, p. 481] provided the following family of rational polytopes in the 0/1-cube with empty integer hull and Chvátal rank n :

$$P_n = \left\{ x \in \mathbb{R}^n \mid \sum_{j \in J} x_j + \sum_{j \notin J} (1 - x_j) \geq \frac{1}{2}, \text{ for all } J \subseteq \{1, \dots, n\} \right\}. \quad (2)$$

The polytopes in this example have exponentially many inequalities, and this indeed has to be the case.

Proposition 1. *Let $P \subseteq [0, 1]^n$ be a polytope in the 0/1-cube with $P_I = \emptyset$ and $\text{rank}(P) = n$. Any inequality description of P has at least 2^n inequalities.*

Proof. For a polytope $P \subseteq \mathbb{R}^n$ and for some $i \in \{1, \dots, n\}$ and $\ell \in \{0, 1\}$ let $P_i^\ell \subseteq \mathbb{R}^{n-1}$ be the polytope defined by

$$P_i^\ell = \{x \in [0, 1]^{n-1} \mid (x_1, \dots, x_{i-1}, \ell, x_{i+1}, \dots, x_n)^T \in P\}.$$

Notice that, if P is contained in a facet $(x_i = \ell)$ of $[0, 1]^n$ for some $\ell \in \{0, 1\}$ and some $i \in \{1, \dots, n\}$, then the Chvátal rank of P is the Chvátal rank of P_i^ℓ .

We will prove now that any one-dimensional face F_1 of the cube satisfies $F_1 \cap P \neq \emptyset$. We proceed by induction on n .

If $n = 1$, this is definitely true since P is not empty and since F_1 is the cube itself. For $n > 1$, observe that any one-dimensional face F_1 of the cube lies in a facet $(x_i = \ell)$ of the cube, for some $\ell \in \{0, 1\}$ and for some $i \in \{1, \dots, n\}$. Since P has Chvátal rank n it follows that $\tilde{P} = (x_i = \ell) \cap P$ has Chvátal rank $n - 1$. If the Chvátal rank of $\tilde{P}$ was less than that, P would vanish after $n - 1$ steps. It follows by induction that $(F_1)_i^\ell \cap \tilde{P}_i^\ell \neq \emptyset$, thus $F_1 \cap P \neq \emptyset$.

Now, each 0/1-point has to be cut off from P by some inequality, as $P_I = \emptyset$. If an inequality $cx \leq \delta$ cuts off two different 0/1-points simultaneously, then it must also cut off a 1-dimensional face of $[0, 1]^n$. Because of our previous observation this is not possible, and hence there is at least one inequality for each 0/1-point which cuts off only this point. Since there are 2^n different 0/1-points in the cube, the claim follows. $\square$

We close this section by introducing some further notation. The ℓ_∞ -norm $\|c\|_\infty$ of a vector $c \in \mathbb{R}^n$ is the largest absolute value of its entries, $\|c\|_\infty = \max\{|c_i| \mid i = 1, \dots, n\}$. The ℓ_1 -norm $\|c\|_1$ of c is the sum $\|c\|_1 = \sum_{i=1}^n |c_i|$. We define the function $\lg : \mathbb{N} \rightarrow \mathbb{N}$ as

$$\lg n = \begin{cases} 1 & \text{if } n = 0 \\ 1 + \lfloor \log_2(n) \rfloor & \text{if } n > 0 \end{cases}$$

where $\lfloor y \rfloor$ denotes the largest integer smaller than or equal to y . Note that $\lg n$ is the number of bits in the binary representation of n . For a vector $x \in \mathbb{R}^n$, $\lfloor x \rfloor$ denotes the vector obtained by component-wise application of $\lfloor \cdot \rfloor$.

3 A New Upper Bound on the Chvátal Rank

We call a vector c *saturated* with respect to a polytope P , if $\max\{cx \mid x \in P\} = \max\{cx \mid x \in P_I\}$. If $Ax \leq b$ is an inequality description of P_I , then $P = P_I$ if and only if each row vector of A is saturated w.r.t. P . In [6], it is shown that an integral vector $c \in \mathbb{Z}^n$ is saturated after at most $n^2 \lg \|c\|_\infty$ steps of the Gomory-Chvátal procedure. Since each 0/1-polytope has a representation $Ax \leq b$ with $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ such that each absolute value of an entry in A is bounded by $n^{1/2}$ (see, e.g., [33]), the known bound of $O(n^3 \lg n)$ follows. One drawback in this proof is that faces of P which do not contain 0/1-points are taken to have worst case behavior n . The following observation is crucial to derive a better bound.

Lemma 4. *Let $cx \leq \alpha$ be valid for P_I and $cx \leq \gamma$ be valid for P , where $\alpha \leq \gamma$, $\alpha, \gamma \in \mathbb{Z}$ and $c \in \mathbb{Z}^n$. If, for each $\beta \in \mathbb{R}$, $\beta > \alpha$, the polytope $F_\beta = P \cap (cx = \beta)$ does not intersect with two opposite facets of the 0/1-cube, then the depth of $cx \leq \alpha$ is at most $2(\gamma - \alpha)$.*

Proof. Notice that $F'_\beta = \emptyset$ for each $\beta > \alpha$. The proof is by induction on $\gamma - \alpha$.

If $\alpha = \gamma$, there is nothing to prove. So let $\gamma - \alpha > 0$. Since $F'_\gamma = \emptyset$, Lemma 1 implies that $cx \leq \gamma - \epsilon$ is valid for P' for some $\epsilon > 0$ and thus the inequality $cx \leq \gamma - 1$ is valid for $P^{(2)}$. $\square$

Proposition 2. *Let P be a rational polytope in the n -dimensional 0/1-cube. Any integral vector $c \in \mathbb{Z}^n$ is saturated w.r.t. $P^{(t)}$, for any $t \geq 2(n^2 + n \lg \|c\|_\infty)$.*

Proof. We can assume that $c \geq 0$ holds and that $P_I \neq \emptyset$. (It is shown in [6] that polytopes with empty integer hull have Chvátal rank at most n .) The proof is by induction on n and $\lg \|c\|_\infty$. The claim holds for $n = 1, 2$ since the Chvátal rank of a polytope in the 1- or 2-dimensional 0/1-cube is at most 4.

So let $n > 2$. If $\lg(\|c\|_\infty) = 1$, then the claim follows, e.g., from Theorem 3 below. So let $\lg \|c\|_\infty > 1$. Write $c = 2c_1 + c_2$, where $c_1 = \lfloor c/2 \rfloor$ and $c_2 \in \{0, 1\}^n$. By induction, it takes at most $2(n^2 + n \lg \|c_1\|_\infty) = 2(n^2 + n \lg \|c\|_\infty) - 2n$

iterations of the Gomory-Chvátal procedure until c_1 is saturated. Let $k = 2(n^2 + n \lg \|c\|_\infty) - 2n$.

Let $\alpha = \max\{cx \mid x \in P_I\}$ and $\gamma = \max\{cx \mid x \in P^{(k)}\}$. The integrality gap $\gamma - \alpha$ is at most n . This can be seen as follows. Choose $\hat{x} \in P^{(k)}$ with $c\hat{x} = \gamma$ and let $x_I \in P_I$ satisfy $c_1 x_I = \max\{c_1 x \mid x \in P^{(k)}\}$. One can choose x_I out of P_I since c_1 is saturated w.r.t. $P^{(k)}$. It follows that

$$\gamma - \alpha \leq c(\hat{x} - x_I) = 2c_1(\hat{x} - x_I) + c_2(\hat{x} - x_I) \leq n.$$

Consider now an arbitrary fixing of an arbitrary variable x_i to a specific value ℓ , $\ell \in \{0, 1\}$. The result is the polytope

$$P_i^\ell = \{x \in [0, 1]^{n-1} \mid (x_1, \dots, x_{i-1}, \ell, x_{i+1}, \dots, x_n)^T \in P\}$$

in the $(n-1)$ -dimensional 0/1-cube for which, by the induction hypothesis, the vector $\tilde{c}_i = (c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_n)$ is saturated after at most

$$2((n-1)^2 + (n-1) \lg \|\tilde{c}_i\|_\infty) \leq 2(n^2 + n \lg \|c\|_\infty) - 2n$$

iterations.

It follows that

$$\alpha - \ell c_i \geq \max\{\tilde{c}_i x \mid x \in (P_i^\ell)^{(k)}\} = \max\{\tilde{c}_i x \mid x \in (P_i^\ell)_I\}.$$

If $\beta > \alpha$, then $(cx = \beta) \cap P^{(k)}$ cannot intersect with a facet of the cube, since a point in $(cx = \beta) \cap P^{(k)} \cap (x_i = \ell)$, $\ell \in \{0, 1\}$, has to satisfy $cx \leq \alpha$.

With Lemma 4, after $2n$ more iterations of the Gomory-Chvátal procedure, c is saturated, which altogether happens after $2(n^2 + n \lg \|c\|_\infty)$ iterations. $\square$

We conclude this section with a new upper bound on the Chvátal rank.

Theorem 2. *The Chvátal rank of a polytope in the n -dimensional 0/1-cube is $O(n^2 \log n)$.*

Proof. Each polytope Q in the 0/1-cube has a rational weakening P . The integral 0/1-polytope P_I can be described by a system of integral inequalities $P_I = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ with $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ such that each absolute value of an entry in A is bounded by $n^{n/2}$. We estimate the number of Gomory-Chvátal steps until all row-vectors of A are saturated. Proposition 2 implies that those row-vectors are saturated after at most $2(n^2 + n \lg n^{n/2}) \leq 3n^2 \lg n$ steps. $\square$

4 A Different Upper Bound on the Depth

In this section we show that any inequality $cx \leq \delta$, which is valid for the integer hull of a polytope P in the n -dimensional 0/1-cube, has depth at most $n + \|c\|_1$ w.r.t. P .

We start by recalling some useful properties of monotone polyhedra, prove then that the Gomory-Chvátal operation is compliant with unimodular transformations, and eventually reduce the general case to the depth of inequalities over monotone polytopes via a special unimodular transformation.

4.1 Monotone Polyhedra

A nonempty polyhedron $P \subseteq \mathbb{R}_{\geq 0}^n$ is called *monotone* if $x \in P$ and $0 \leq y \leq x$ imply $y \in P$. Hammer, Johnson, and Peled [26] observed that a polyhedron P is monotone if and only if P can be described by a system $x \geq 0$, $Ax \leq b$ with $A, b \geq 0$.

The next statements are proved in [27] and [13, p. 494]. We include a proof of Lemma 6 for the sake of completeness.

Lemma 5. *If P is a monotone polyhedron, then P' is monotone as well.*

Lemma 6. *Let P be a monotone polytope in the 0/1-cube and let $w x \leq \delta$, $w \in \mathbb{Z}^n$, be valid for P_I . Then $w x \leq \delta$ has depth at most $\|w\|_1 - \delta$.*

Proof. The proof is by induction on $\|w\|_1$. If $\|w\|_1 = 0$, the claim follows trivially. W.l.o.g., we can assume that $w \geq 0$ holds. Let $\gamma = \max\{w x \mid x \in P\}$ and let $J = \{j \mid w_j > 0\}$. If $\max\{\sum_{j \in J} x_j \mid x \in P\} = |J|$, then, since P is monotone, $\hat{x}$ with

$$\hat{x}_i = \begin{cases} 1 & \text{if } i \in J, \\ 0 & \text{otherwise} \end{cases}$$

is in P . Also $w \hat{x} = \gamma$ must hold. So $\gamma = \delta$ and the claim follows trivially. If $\max\{\sum_{j \in J} x_j \mid x \in P\} < |J|$, then $\sum_{j \in J} x_j \leq |J| - 1$ has depth at most 1. If $\|w\|_1 = 1$ this also implies the claim, so assume $\|w\|_1 \geq 2$. By induction the valid inequalities $w x - x_j \leq \delta$, $j \in J$ have depth at most $\|w\|_1 - \delta - 1$. Adding up the inequalities $w x - x_j \leq \delta$, $j \in J$ and $\sum_{j \in J} x_j \leq |J| - 1$ yields

$$w x \leq \delta + (|J| - 1)/|J|.$$

Rounding down yields $w x \leq \delta$ and the claim follows. $\square$

4.2 Unimodular Transformations

Unimodular transformations and in particular switching operations will play a crucial role to relate the Chvátal rank of arbitrary polytopes in the 0/1-cube to the Chvátal rank of monotone polytopes. In this section, we show that unimodular transformations and the Gomory-Chvátal operation commute.

A *unimodular transformation* is a mapping

$$\begin{aligned} u : \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ x &\mapsto Ux + v, \end{aligned}$$

where $U \in \mathbb{Z}^{n \times n}$ is a unimodular matrix, i.e., $\det(U) = \pm 1$, and $v \in \mathbb{Z}^n$.

Note that u is a bijection. Its inverse is the unimodular transformation $u^{-1}(x) = U^{-1}x - U^{-1}v$. Since $U^{-1} \in \mathbb{Z}^{n \times n}$, u is also a bijection of $\mathbb{Z}^n$.

Consider the rational halfspace $(cx \leq \delta)$, $c \in \mathbb{Z}^n, \delta \in \mathbb{Q}$. The set $u(cx \leq \delta)$ is the rational halfspace

$$\begin{aligned} \{x \in \mathbb{R}^n \mid cu^{-1}(x) \leq \delta\} &= \{x \in \mathbb{R}^n \mid cU^{-1}x \leq \delta + cU^{-1}v\} \\ &= (cU^{-1}x \leq \delta + cU^{-1}v). \end{aligned}$$

Notice that the vector cU^{-1} is also integral. Let S be some subset of $\mathbb{R}^n$. It follows that $(cx \leq \delta) \supseteq S$ if and only if $(cU^{-1}x \leq \delta + cU^{-1}v) \supseteq u(S)$.

Consider now the first elementary closure P' of some polyhedron P ,

$$P' = \bigcap_{\substack{(cx \leq \delta) \supseteq P \\ c \in \mathbb{Z}^n}} (cx \leq \lfloor \delta \rfloor).$$

It follows that

$$u(P') = \bigcap_{\substack{(cx \leq \delta) \supseteq P \\ c \in \mathbb{Z}^n}} (cU^{-1}x \leq \lfloor \delta \rfloor + cU^{-1}v).$$

From this one can derive the next lemma.

Lemma 7. *Let P be a polyhedron and u be a unimodular transformation. Then*

$$u(P') = (u(P))'.$$

Corollary 1. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let $cx \leq \delta$ be a valid inequality for P_I . Let u be a unimodular transformation. The inequality $cx \leq \delta$ is valid for $P^{(k)}$ if and only if $u(cx \leq \delta)$ is valid for $(u(P))^{(k)}$.*

The i -th switching operation is the unimodular transformation

$$\begin{aligned} \pi_i : \quad \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ (x_1, \dots, x_n) &\mapsto (x_1, \dots, x_{i-1}, 1 - x_i, x_{i+1}, \dots, x_n), \end{aligned}$$

It has a representation

$$\begin{aligned} \pi_i : \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ x &\mapsto Ux + e_i, \end{aligned}$$

where U coincides with the identity matrix I_n except for $U_{(i,i)}$ which is -1 . Note that the switching operation is a bijection of $[0, 1]^n$. For the set $(cx \leq \delta)$ one has $\pi_i(cx \leq \delta) = \tilde{c}x \leq \delta - c_i$. Here $\tilde{c}$ coincides with c except for a change of sign in the i -th component.

4.3 The Reduction to Monotone Weakenings

If one wants to examine the depth of a particular inequality with respect to a polytope $P \subseteq [0, 1]^n$, one can apply a series of switching operations until all its coefficients become nonnegative. An inequality with nonnegative coefficients defines a (fractional) 0/1-knapsack polytope K . The depth of this inequality with respect to the convex hull of P and K is then an upper bound on the depth with respect to P . We will show that $\text{conv}(P, K)^{(n)}$ has a monotone weakening in the 0/1-cube.

Lemma 8. *Let $P \subseteq [0, 1]^n$ be a polytope in the 0/1-cube, with $P_I = K_I$, where $K = \{x \mid cx \leq \delta, 0 \leq x \leq 1\}$ and $c \geq 0$. Then, $P^{(n)}$ has a rational, monotone weakening Q in the 0/1-cube.*

Proof. We can assume that P is rational. Let $\hat{x}$ be a 0/1-point which is not contained in P , i.e., $c\hat{x} > \delta$. Let $I = \{i \mid \hat{x}_i = 1\}$. The inequality $\sum_{i \in I} x_i \leq |I|$ is valid for the cube and thus for P . Since $c \geq 0$, the corresponding face $F = \{x \mid \sum_{i \in I} x_i = |I|, x \in P\}$ of P does not contain any 0/1-points. Lemma 2 implies that $\sum_{i \in I} x_i \leq |I| - 1$ is valid for $P^{(n)}$.

Thus, for each 0/1-point $\hat{x}$ which is not in P , there exists a nonnegative rational inequality $a_{\hat{x}}x \leq \gamma_{\hat{x}}$ which is valid for $P^{(n)}$ and which cuts $\hat{x}$ off. Thus

$$\begin{aligned} 0 &\leq x_i \leq 1, \quad i \in \{1, \dots, n\} \\ a_{\hat{x}}x &\leq \gamma_{\hat{x}}, \quad \hat{x} \in \{0, 1\}^n, \hat{x} \notin P \end{aligned}$$

is the desired weakening. $\square$

Theorem 3. *Let $P \subseteq [0, 1]^n$, $P \neq \emptyset$ be a nonempty polytope in the 0/1-cube and let $cx \leq \delta$ be a valid inequality for P_I with $c \in \mathbb{Z}^n$. Then $cx \leq \delta$ has depth at most $n + \|c\|_1$ with respect to P .*

Proof. One can assume that c is nonnegative, since one can apply a series of switching operations. Notice that this can change the right hand side δ , but in the end δ has to be nonnegative since $P \neq \emptyset$. Let $K = \{x \in [0, 1]^n \mid cx \leq \delta\}$ and consider the polytope $Q = \text{conv}(K, P)$. The inequality $cx \leq \delta$ is valid for Q_I and the depth of $cx \leq \delta$ with respect to P is at most the depth of $cx \leq \delta$ with respect to Q . By Lemma 8, $Q^{(n)}$ has a monotone weakening S . The depth of $cx \leq \delta$ with respect to $Q^{(n)}$ is at most the depth of $cx \leq \delta$ with respect to S . But it follows from Lemma 6 that the depth of $cx \leq \delta$ with respect to S is at most $\|c\|_1 - \delta \leq \|c\|_1$. $\square$

5 A New Lower Bound on the Chvátal Rank

To the best of the authors' knowledge, no example of a polytope P in the n -dimensional 0/1-cube with $\text{rank}(P) > n$ has been provided in the literature so far. We now show that $r(n) > (1 + \epsilon)n$, for infinitely many n , where $\epsilon > 0$.

The construction relies on the lower bound result for the fractional stable-set polytope due to Chvátal, Cook, and Hartmann [13].

Let $G = (V, E)$ be a graph on n vertices, $\mathcal{C}$ be the family of all cliques of G , and let $Q \subseteq \mathbb{R}^n$ be the fractional stable set polytope of G defined by the equations

$$\begin{aligned} x(C) &\leq 1 \text{ for all } C \in \mathcal{C}, \\ x_v &\geq 0 \text{ for all } v \in V. \end{aligned} \tag{3}$$

Let $\mathbf{e}$ be the vector of all ones. The following lemma is proved in [13, Proof of Lemma 3.1].

Lemma 9. *Let $k < s$ be positive integers and let G be a graph with n vertices such that every subgraph of G with s vertices is k -colorable. If P is a polyhedron that contains Q_I and the point $u = \frac{1}{k}\mathbf{e}$, then $P^{(j)}$ contains the point $x^j = (\frac{s}{s+k})^j u$.*

Let $\alpha(G)$ be the size of the largest independent subset of the nodes of G . It follows that $\mathbf{e}x \leq \alpha(G)$ is valid for Q_I . One has

$$\mathbf{e}x^j = \frac{n}{k} \left(\frac{s}{s+k} \right)^j \geq \frac{n}{k} e^{-jk/s},$$

and thus x^j does not satisfy the inequality $\mathbf{e}x \leq \alpha(G)$ for all $j < (s/k) \ln \frac{n}{k\alpha(G)}$.

Erdős proved in [17] that for every positive t there exist a positive integer c , a positive number δ and arbitrarily large graphs G with n vertices, cn edges, $\alpha(G) < tn$ and every subgraph of G with at most δn vertices is 3 colorable. One wants that $\ln \frac{n}{k\alpha(G)} \geq 1$ and that s/k grows linearly, so by choosing some $t < 1/(3e)$, $k = 3$ and $s = \lfloor \delta n \rfloor$ one has that x^j does not satisfy the inequality $\mathbf{e}x \leq \alpha(G)$ for all $j < (s/k)$.

We now give the construction. Let P be the polytope that results from the convex hull of P_n defined in (2) and Q . $P_n \subseteq P$ contributes to the fact that $1/2\mathbf{e}$ is in $P^{(n-1)}$ [13, Lemma 7.2]. Thus $x_0 = 1/3\mathbf{e}$ is in $P^{(n-1)}$. Since the convex hull of P is Q_I , it follows from the above discussion that the depth of $\mathbf{e}x \leq \alpha(G)$ with respect to $P^{(n-1)}$ is $\Omega(n)$. Thus the depth of $\mathbf{e}x \leq \alpha(G)$ is at least $(n-1) + \Omega(n) \geq (1+\epsilon)n$ for infinitely many n , where $\epsilon > 0$.

Acknowledgments

The authors are grateful to Alexander Bockmayr, Volker Priebe, and Günter Ziegler for helpful comments on an earlier version of this paper.

References

- [1] N. Alon and V. H. Vu. Anti-Hadamard matrices, coin weighing, threshold gates, and indecomposable hypergraphs. *Journal of Combinatorial Theory*, 79A:133–160, 1997.

- [2] E. Balas, S. Ceria, G. Cornuéjols, and N. R. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19:1–9, 1996.
- [3] E. Balas and M. J. Saltzman. Facets of the three-index assignment polytope. *Discrete Applied Mathematics*, 23:201–229, 1989.
- [4] F. Barahona, M. Grötschel, and A. R. Mahjoub. Facets of the bipartite subgraph polytope. *Mathematics of Operations Research*, 10:340–358, 1985.
- [5] A. Bockmayr and F. Eisenbrand. On the Chvátal rank of polytopes in the 0/1 cube. Research Report MPI-I-97-2-009, Max-Planck-Institut für Informatik, September 1997.
- [6] A. Bockmayr, F. Eisenbrand, M. E. Hartmann, and A. S. Schulz. On the Chvátal rank of polytopes in the 0/1 cube. Technical Report 616, Technical University of Berlin, Department of Mathematics, December 1998.
- [7] M. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *Journal of Symbolic Logic*, 62:708–728, 1997.
- [8] S. C. Boyd and W. H. Cunningham. Small travelling salesman polytopes. *Mathematics of Operations Research*, 16:259–271, 1991.
- [9] S. C. Boyd, W. H. Cunningham, M. Queyranne, and Y. Wang. Ladders for travelling salesmen. *SIAM Journal on Optimization*, 5:408–420, 1995.
- [10] S. C. Boyd and W. R. Pulleyblank. Optimizing over the subtour polytope of the travelling salesman problem. *Mathematical Programming*, 49:163–187, 1991.
- [11] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.
- [12] V. Chvátal. Flip-flops in hypohamiltonian graphs. *Canadian Mathematical Bulletin*, 16:33–41, 1973.
- [13] V. Chvátal, W. Cook, and M. E. Hartmann. On cutting-plane proofs in combinatorial optimization. *Linear Algebra and its Applications*, 114/115:455–499, 1989.
- [14] W. Cook, C. R. Coullard, and Gy. Turán. On the complexity of cutting plane proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.
- [15] W. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley, 1998.
- [16] F. Eisenbrand. A note on the membership problem for the first elementary closure of a polyhedron. Technical Report 605, Technical University of Berlin, Department of Mathematics, November 1998. To appear in *Combinatorica*.
- [17] P. Erdős. On circuits and subgraphs of chromatic graphs. *Mathematika*, 9:170–175, 1962.
- [18] M. Fischetti. Three facet lifting theorems for the asymmetric traveling salesman polytope. In E. Balas, G. Cornuéjols, and R. Kannan, editors, *Integer Programming and Combinatorial Optimization*, pages 260–273. Proceedings of the 2nd IPCO Conference, 1992.
- [19] T. Fleiner, V. Kaibel, and G. Rote. Upper bounds on the maximal number of facets of 0/1-polytopes. Technical Report 98-327, University of Cologne, Department of Computer Science, 1998. To appear in *European Journal of Combinatorics*.
- [20] R. Giles and L. E. Trotter. On stable set polyhedra for $K_{1,3}$ -free graphs. *Journal of Combinatorial Theory*, 31:313–326, 1981.
- [21] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.
- [22] R. E. Gomory. An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, 1963.

- [23] M. Grötschel and M. W. Padberg. Polyhedral theory. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 251–305. John Wiley, 1985.
- [24] M. Grötschel and W. R. Pulleyblank. Clique tree inequalities and the symmetric travelling salesman problem. *Mathematics of Operations Research*, 11:537–569, 1986.
- [25] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.
- [26] P. L. Hammer, E. Johnson, and U. N. Peled. Facets of regular 0-1 polytopes. *Mathematical Programming*, 8:179–206, 1975.
- [27] M. E. Hartmann. Cutting planes and the complexity of the integer hull. Technical Report 819, School of Operations Research and Industrial Engineering, Cornell University, September 1988.
- [28] M. E. Hartmann. Personal communication, March 1998.
- [29] M. E. Hartmann, M. Queyranne, and Y. Wang. On the Chvátal rank of certain inequalities. This volume, 1999.
- [30] R. Impagliazzo, T. Pitassi, and A. Urquhart. Upper and lower bound for tree-like cutting plane proofs. In *Proc. Logic in Computer Science, LICS'94, Paris*, 1994.
- [31] U. H. Kortenkamp, J. Richter-Gebert, A. Sarangarajan, and G. M. Ziegler. Extremal properties of 0/1-polytopes. *Discrete and Computational Geometry*, 17:439–448, 1997.
- [32] D. Naddef. The Hirsch conjecture is true for (0,1)-polytopes. *Mathematical Programming*, 45:109–110, 1989.
- [33] M. W. Padberg and M. Grötschel. Polyhedral computations. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 307–360. John Wiley, 1985.
- [34] P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62:981–988, 1997.
- [35] W. R. Pulleyblank. Polyhedral combinatorics. In G. L. Nemhauser, A. H. G. Rinnoy Kan, and M. J. Todd, editors, *Optimization*, Volume 1 of *Handbooks in Operations Research and Management Science*, Chapter V, pages 371–446. Elsevier, 1989.
- [36] A. Schrijver. On cutting planes. *Annals of Discrete Mathematics*, 9:291–296, 1980.
- [37] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley, 1986.
- [38] A. S. Schulz. *Polytopes and Scheduling*. PhD thesis, Technical University of Berlin, Berlin, Germany, 1996.
- [39] A. S. Schulz. A simple proof that the Chvátal rank of polytopes in the 0/1-cube is small. Unpublished manuscript, September 1997.
- [40] A. S. Schulz, R. Weismantel, and G. M. Ziegler. An optimization problem is ten problems. In preparation.

Universally Maximum Flow with Piecewise-Constant Capacities

Lisa Fleischer

Department of Industrial Engineering and Operations Research
Columbia University, New York, NY 10027
`lisa@ieor.columbia.edu`

Abstract. The *maximum dynamic flow problem* generalizes the standard maximum flow problem by introducing time. The object is to send as much flow from source to sink in T time units as possible, where capacities are interpreted as an upper bound on the rate of flow entering an arc. A related problem is the *universally maximum flow*, which is to send a flow from source to sink that maximizes the amount of flow arriving at the sink by time t simultaneously for all $t \leq T$. We consider a further generalization of this problem that allows arc and node capacities to change over time. In particular, given a network with arc and node capacities that are piecewise constant functions of time with at most k breakpoints, and a time bound T , we show how to compute a flow that maximizes the amount of flow reaching the sink in all time intervals $(0, t]$ simultaneously for all $0 < t \leq T$, in $O(k^2 mn \log(kn^2/m))$ time. The best previous algorithm requires $O(nk)$ maximum flow computations on a network with $(m+n)k$ arcs and nk nodes.

1 Introduction

In the 1960's, Ford and Fulkerson introduced dynamic network flows to include time in the standard network flow model. Since then, dynamic network flows have been used widely to model network-structured, decision-making problems over time: problems in electronic communication, production and distribution, economic planning, cash flow, job scheduling, and transportation. For examples, see the surveys of Aronson [4] and Powell, et al. [20].

The *maximum dynamic flow problem* generalizes the standard maximum flow problem by introducing time. A standard network consists of a set of nodes V and a set of arcs E which is a subset of $V \times V$. The capacity function from the arcs to the real numbers bounds the amount of flow allowed on each arc. In a *dynamic network*, the capacity function u limits the rate of flow into an arc. In addition, a dynamic network has a transit-time vector ϱ associated with the arcs. The transit time of an arc is the amount of time it takes for flow to travel from one end of the arc to the other.

Ford and Fulkerson [7] consider the dynamic maximum flow problem: given a dynamic network with a specified source and sink, determine the maximum amount of flow that can be sent from source to sink in T time units. They show

that the problem can be solved in polynomial time by using information obtained from a minimum cost flow computation in a related network of comparable size.

A generalization of dynamic maximum flows, a *universally maximum flow* is a flow that simultaneously maximizes the amount of flow reaching a specified sink node d , sent from a specified source node s , by time t , for all $0 < t \leq T$. Wilkinson [24] and Minieka [14] showed that such a flow exists, and they both provide algorithms to solve this problem, but these do not run in polynomial time. There is no known polynomial time algorithm that solves the universally maximum flow problem. It is not even known if the optimal solution is polynomial in the size of the input. Hoppe and Tardos [11,12] present an $O(m\epsilon^{-1}(m + n \log n) \log U)$ algorithm that computes a dynamic flow with the property that the quantity of flow reaching the sink by time t is within $(1 - \epsilon)$ of the maximum possible for all $0 \leq t \leq T$.

Another generalization of the maximum dynamic flow problem is the problem with multiple sources and sinks. The *dynamic transshipment problem* is a single commodity flow problem that asks to send specified supplies located at source nodes to satisfy specified demands at sink nodes within a given time bound T . A *universally quickest transshipment* is a dynamic flow that satisfies as much demand as possible in every interval of time $(0, t]$ for $0 < t \leq T$. If there is more than one sink, a universally quickest transshipment may not exist, so the universally quickest transshipment problem refers to a dynamic transshipment problem with one sink only.

One interesting special case of dynamic flow problems is when all transit times are zero. In this setting, the universally maximum flow problem is solved by sending flow at the rate of a maximum flow in the static network at every moment of time in $[0, T]$. (The *static network* is the interpretation of the dynamic network as a standard network. Transit times are ignored, and capacities are interpreted as bound on the total flow allowed on an arc.) A universally quickest transshipment with zero transit times has a more complicated structure. This problem models the situation when the supplies and demands exceed network capacity, and it is necessary to send flow over time, in rounds. Unlike standard flow problems with multiple sources and sinks, this problem cannot be modeled as an equivalent $s - t$ dynamic maximum flow problem. The reason such a transformation does not work here is because the arc capacities are upper bounds on the rate of flow per time period, and do not correspond to the total amount of flow an arc may carry throughout an interval. The dynamic transshipment problem with zero transit times is discussed in [6,10,15,23]. Hajek and Ogier [10] describe the first polynomial time algorithm to find a universally quickest transshipment for a dynamic network with all zero transit times. Their algorithm uses n maximum flow computations on the underlying static network. This is improved in [6] with an algorithm that solves this problem in the same asymptotic time as a preflow-push maximum flow algorithm.

All of the above mentioned problems may be generalized to allow storage of flow at the nodes. In this setting, there may be node storage capacities that limit the total excess flow allowed at a node. Flow deficit is not allowed. In

the above mentioned problems, even when node excess is allowed, there always exists an optimal solution that does not use any node storage, and the algorithms mentioned above find such a solution. For the universally quickest transshipment with piecewise-constant capacity functions, the subject of this paper, this will not be the case: node storage will be used in an optimal solution, and the amount of available storage will affect the solution.

Many other generalizations of this problem have been considered in the literature, however few other generalizations are known to be polynomial-time solvable. One exception is the integer dynamic transshipment problem with general transit times, for which Hoppe and Tardos [13] give the only polynomial time algorithm known to solve this problem. Their algorithm repeatedly calls an oracle to minimize submodular functions. Orlin [17] provides polynomial time algorithms for some infinite horizon problems, where the objective function is related to average throughput. Burkard, et al. [5] find the minimum feasible time to send specified flow from source to sink faster than incorporating the Ford-Fulkerson algorithm in a binary search framework. Most other work has focussed on characterizing the structure of problems with more general capacity functions, and determining when optimal solutions exist [1,2,3,18,19,21,22]. None of these problems are known to be polynomial time solvable. Among the earlier work, Anderson, Nash, and Philpott [2] consider the problem of finding a maximum flow in a network with zero transit times, and with capacities and node storage that are much more general functions of time, and develop a duality theory for continuous network flows.

We consider the generalization of the zero transit time, universally maximum flow problem that allows both arc capacities and node storage capacities to be piecewise-constant functions on $[0, T]$ with at most k breakpoints. As with the general universally maximum flow problem, it is not clear a universally maximum flow exists in these circumstances. Ogier [16] proves it does, and provides a polynomial time algorithm that finds a universally maximum flow with at most kn breakpoints. These breakpoints can be computed with nk maximum flow computations on a network with nk vertices and $(m + n)k$ arcs. After the breakpoints are computed, additional maximum flow computations on the same network are used to calculate the flow between two breakpoints. Thus the total run time of Ogier's algorithm is determined by the time to solve $O(nk)$ maximum flow problems on a network with nk vertices and $(n + m)k$ arcs. As Ogier [16] demonstrates, this problem also generalizes the universally quickest transshipment problem.

The main contributions of this paper is to a) recognize that these problems can be solved by solving a parametric maximum flow problem on a suitably defined graph and b) generalize the parametric maximum flow algorithm of Gallo, Grigoriadis, and Tarjan [8] to fit the needs of this problem. The end result is that all the computations described in [16] can be performed in the same asymptotic time as one preflow-push maximum flow computation on a network with nk vertices and $(n + m)k$ arcs. This improves the previous strongly polynomial run time by a factor of $O(nk)$.

1.1 The Model

The universally maximum flow problem with piecewise constant capacity functions (UMFP) is defined on a network with n nodes and m arcs, along with a nonnegative, left-continuous, piecewise-constant capacity function $u : (E \cup V) \times (0, T] \rightarrow \mathbf{Z}_+ \cup \{0\}$. When referring to the capacity of a particular arc e or a particular node i at a particular time t , we use the notation $u_e(t)$ or $u_i(t)$. The vector of all capacities at time t is simply denoted $u(t)$, the vector of just arc capacities is denoted $u_E(t)$, and the vector of node capacities is denoted $u_V(t)$. For $A \subset V$, we use the notation $u_{A, \overline{A}}(t)$ to denote $\sum_{i \in A, j \in \overline{A}} u_{ij}(t)$, and similarly for a flow function $x(t)$.

For convenience of notation, we assume that the breakpoints of the capacity functions are exactly one unit apart. If this is not the case, we can scale time between each breakpoint to create an equivalent problem with this property. That is, if the time between two breakpoints is τ , then we create an equivalent problem by changing this to 1 and multiplying all capacities in this interval by τ . Any flow that solves this modified problem can be transformed back to a solution to the original problem by dividing the rate of flow sent in the new time interval by τ and sending it over an interval that is τ times longer. In such a manner, we scale T so that it equals k . Thus, for the remainder of the paper, we assume our time bound is k , all k breakpoints are in the set $\{1, 2, \dots, k\}$, the domain of u is $(E \cup V) \times (0, k]$, and $u(\tau) = u(\lceil \tau \rceil)$ for all $0 \leq \tau \leq k$.

Define x to be a feasible dynamic flow if x_e is a Lebesgue-measurable function on $(0, k]$ for all $e \in E$, x obeys the *antisymmetry constraints* $x_e = -x_{\overline{e}}$ for all $e \in E$ (where $\overline{e}$ is the reverse of arc e) and the *arc capacity constraints* $x_e(t) \leq u_e(t)$ for all $e \in E$ and all $t \in (0, k]$; and the excess at each node, defined by

$$p_j(\tau) := \int_0^\tau \sum_{i \in V} x_{ij}(t) dt,$$

obeys the *node capacity constraint* $0 \leq p_j(t) \leq u_j(t)$ for all $j \in V - \{s, d\}$, and all $0 \leq t \leq k$. Denote the set of feasible dynamic flows by D . The value of flow x at time τ , denoted $v_\tau(x)$, is the total flow reaching the sink by time τ . That is,

$$v_\tau(x) := \int_0^\tau x_{N,d}(t) dt.$$

In this paper, we will use a series of related time-expanded graphs. A *time-expanded network* of $\mathcal{N}$ is a directed graph that contains a copy of $\mathcal{N}$ for each time unit, and *holdover arcs* from a copy of a node at time θ to the copy of the same node at time $\theta + 1$ for $\theta \in \mathbf{Z}$. See Figure 1. Let G_τ be the time-expanded graph on $k + 1$ copies of $\mathcal{N}$, labeled $\mathcal{N}_0, \dots, \mathcal{N}_k$. Let $i(\theta)$ be the copy of vertex i in copy $\mathcal{N}_\theta$. Assign the arcs in $\mathcal{N}_\theta$ capacities $u_E(\theta)$ for $\theta \in \{0, \dots, \lceil \tau \rceil\}$, and 0 otherwise. Assign holdover arcs leaving $\mathcal{N}_\theta$ capacities equal to $u_V(\theta + 1)$ for $\theta = 0$ to $\lceil \tau \rceil$, and zero otherwise. By our scaling assumption, there are exactly $k + 1$ unique graphs G_τ , one for each $\theta \in \{0, 1, \dots, k\}$, such that for all $0 \leq \tau \leq k$, $G_\tau = G_{\lceil \tau \rceil}$.

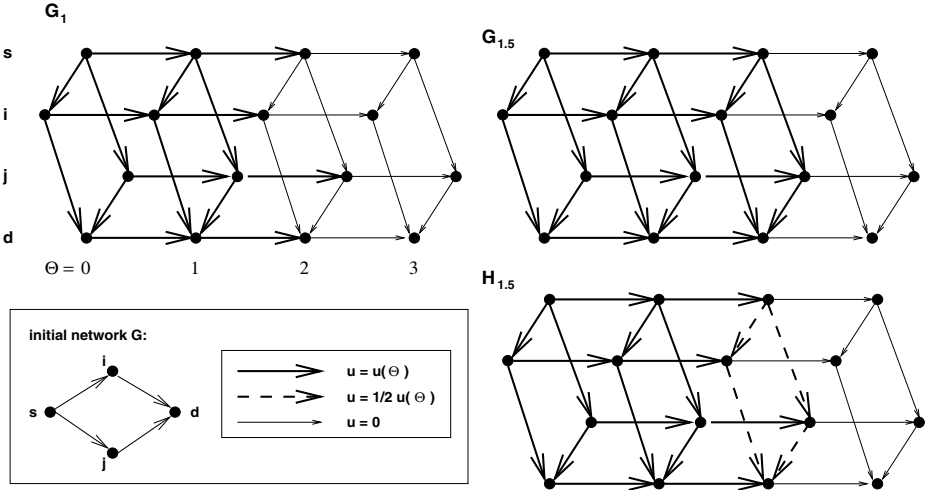


Fig. 1. Examples of time-expanded graphs G_τ and H_τ for $\tau = 1, 1.5$.

1.2 Optimality Conditions

A τ -maximum flow is a feasible dynamic flow x that maximizes $v_\tau(x)$. As in the static flow setting, we can define a notion of dynamic cut that proves the optimality of a τ -maximum flow. Dynamic cuts can be interpreted as cuts in a modified time-expanded network of $\mathcal{N}$. A *dynamic cut* is a function C that maps $[0, k]$ to subsets of vertices of the network $\mathcal{N}$ such that $C(\alpha) = C(\beta)$ for all $\alpha, \beta \in (\theta, \theta + 1]$, $\theta \in \{0, 1, \dots, k-1\}$, and $C(t) \cap \{s, d\} = \{d\}$ for all $t \in (0, k]$. Note that this use of cut is the complement of what is traditionally defined as a cut. Let $\mathcal{C} := \{C \mid C \text{ is a dynamic cut}\}$. Since the range of a dynamic cut is a finite set, and, over the domain $[0, k]$, each dynamic cut can change only k times (each is uniform over each interval $(\theta, \theta + 1]$), $\mathcal{C}$ is a finite set. The value of dynamic cut C at time τ is determined by arc capacities (the first term) and node capacities (the second term) and is expressed as

$$w_\tau(C) = \int_0^\tau u_{\overline{C(t)}C(t)}(t)dt + \sum_{\theta=1}^{\lceil \tau \rceil - 1} \sum_{j \in C(\theta+1) \cap \overline{C(\theta)}} u_j(\theta). \quad (1)$$

A dynamic cut can be interpreted as a set of arcs entering a cut in the time-expanded network. This will be made more explicit following Theorem 1. This theorem is comparable to strong duality in standard network flows. The proof of even the weak duality case is a little more involved than the corresponding static version and is omitted. A slightly weaker statement is proved by Anderson, Nash, and Philpott [1], but for more general capacity functions.

Theorem 1 (Ogier [16]). For all τ , $\min_{C \in \mathcal{C}} w_\tau(C) = \max_{x \in D} v_\tau(x)$.

Corollary 1. *If $\theta \in \{1, 2, \dots, k\}$, then a maximum flow in the time-expanded network G_θ corresponds to a θ -maximum flow of the same value, where the flow on arc $e(\theta)$ in G_θ is precisely the flow on this arc in the θ -maximum flow over the interval $(\theta, \theta + 1]$.*

Proof. A maximum flow in G_θ gives rise to a minimum cut R_θ that contains all copies of the sink node in G_θ , and no copies of the source node. Letting $C(\tau) = C(\lceil \tau \rceil) := \mathcal{N}_{\lceil \tau \rceil} \cap R_\theta$ for all $\tau \in (0, k]$, we have that C is a dynamic cut of value equal to the maximum flow in G_θ , and hence equal to the value of the corresponding dynamic flow described in the corollary.

We extend the series of time-expanded graphs G_τ to a more continuous version H_τ , so that we may relate τ -maximum flows to τ -minimum cuts for $\tau \notin \{1, 2, \dots, k\}$. For each real $0 \leq \tau \leq k$, we define H_τ to be a graph on the same vertex set and support as G_τ , but with a slightly different capacity function. In H_τ the capacities of the arcs in $\mathcal{N}_{\lceil \tau \rceil + 1}$ are multiplied by $\tau - \lfloor \tau \rfloor$, which is always less than one. The capacities of all other arcs equal the capacities of the arcs in G_τ . Denote the capacity function of H_τ by $\tilde{w}_\tau$. Note that $H_\theta = G_\theta$ for all $\theta \in \{0, 1, \dots, k\}$. H_τ and G_τ are depicted in Figure 1.

The value of a dynamic cut at time τ corresponds to the value of the arcs entering the sink side of a cut in H_τ . We make this explicit as follows. Denote by V_τ^* the vertex set of H_τ (and G_τ). Since the vertex sets of each H_τ and G_τ are the same for all τ , we will denote this V^* when the context is clear. A *cut* in H_τ is a set $R \subset V^*$ such that R contains all copies of the sink node, and no copies of the source. That is, if s_θ is the copy of s and d_θ is the copy of d in N_θ for $\theta \in \{0, 1, \dots, k\}$, then $R \cap \bigcup_{\theta=0}^k \{s_\theta, d_\theta\} = \bigcup_{\theta=0}^k \{d_\theta\}$. See Figure 2.

For a set $R \subset V^*$, define $\tilde{w}_\tau(R)$ to be the sum of capacities of arcs in H_τ entering R . We can equate each dynamic cut $C \in \mathcal{C}$ with a cut R such that $w_\tau(C) = \tilde{w}_\tau(R)$ by setting $R \cap \mathcal{N}_\theta = C(\theta)$ for $\theta \in \{0, 1, \dots, k\}$. Define $R(\theta) := R \cap \mathcal{N}_\theta$ for $\theta \in \{0, 1, \dots, k\}$. The following corollary follows easily from the above discussions.

Corollary 2. $\max_{x \in D} v_\tau(x) = \min_{R \subset V^*} \tilde{w}_\tau(R)$ and a maximum flow in H_τ yields a corresponding τ -maximum flow, where the correspondence is as in Corollary 1.

By the submodularity of cut functions in a graph, the non-empty intersection of minimum cuts is a minimum cut. Thus, there is a minimum cut in H_τ with smallest sink-side. Let R_τ be the minimum cut such that $R_\tau \subseteq R$ for all R such that $\tilde{w}_\tau(R) = \tilde{w}_\tau(R_\tau)$. For examples, see Figure 2.

Lemma 1 (Ogier [16]). *For all $0 \leq \tau < \sigma \leq k$, $R_\tau \subseteq R_\sigma$, and R_τ is a left continuous function of τ .*

Let W be the set of breakpoints of R_τ as a function of τ . That is, W is the set of values τ such that $R_\tau \neq R_{\tau+}$, where $R_{\tau+}$ indicates the limit of R_σ as $\sigma \downarrow \tau$. Ogier's main contribution is to prove that there is an optimal solution to UMFP that is uniform on the intervals between successive breakpoints of W ,

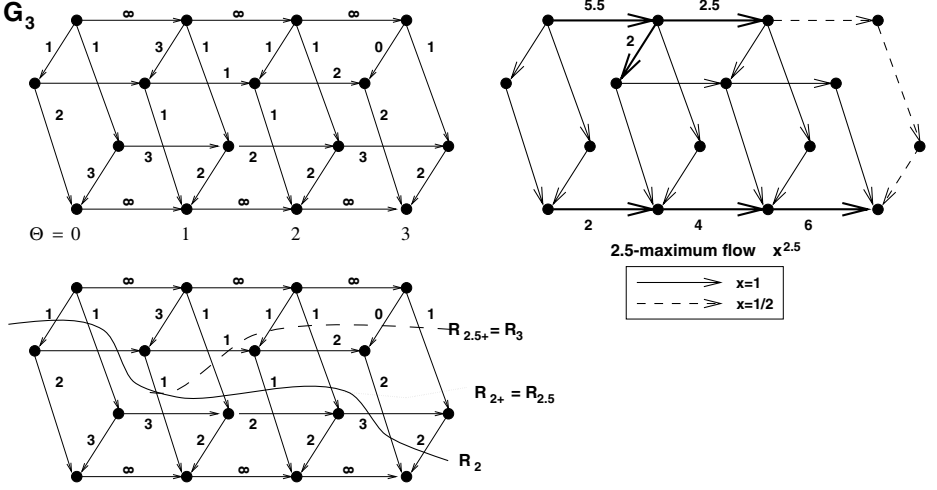


Fig. 2. Top left: The time expanded graph G_k for $k = 3$. Bottom left: Cuts R_τ for $\tau = 2, 2.5, 3$. Top right: A τ -maximum flow x^τ for $\tau = 2.5$.

and that such an optimal flow can be defined piecemeal by special τ -maximum flows [16].

Let x^τ be a τ -maximum flow that is constant on each interval $(\theta - 1, \theta]$ for $\theta \in \{1, 2, \dots, \lfloor \tau \rfloor\}$ and also constant on interval $(\lfloor \tau \rfloor, \tau]$. In addition, the excess function p^τ of x^τ satisfies $p_j^\tau(\tau) = 0$ for all $j \in V - \{s, d\}$. That such a flow exists is implied by Corollary 2 : Compute a maximum flow in H_τ . This flow saturates the arcs entering R_τ , hence has the value of a τ -maximum flow. It does not use any node storage arc leaving $\mathcal{N}(\lceil \tau \rceil)$, hence in the dynamic setting, completes by time τ . (See also Figure 2.) Define x^0 to be the dynamic flow such that for all t , $f = x^0(t)$ is a maximum flow for static network $\mathcal{N}$ with capacities $u(t)$, sources $R_t(\lceil t \rceil)$, and sink d . Theorem 2 below extends the static flow concept of complementary slackness to UMFP.

For each node j of the dynamic network $\mathcal{N}$, define

$$q_j(t) := \begin{cases} \max\{\tau \mid \tau \geq t, j(\lceil \tau \rceil) \in \overline{R_\tau}\}, & \text{if this set is non-empty,} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

In words, $q_j(t)$ is the largest value of τ for which there is no path from $j(\lceil \tau \rceil)$ to $d(\lceil \tau \rceil)$ in the residual network of a τ -maximum flow. Thus if $q_i(t) > q_j(t)$ then there is a τ such that for all x^σ , $\sigma > \tau$, if $i(\lceil \tau \rceil)$ has a residual path to $d(\lceil \sigma \rceil)$, then $j(\lceil \tau \rceil)$ has a residual path to $d(\lceil \sigma \rceil)$, and there is some such σ for which $i(\lceil \tau \rceil)$ does not have a residual path to $d(\lceil \sigma \rceil)$, when $j(\lceil \tau \rceil)$ does. Since R_τ is left-continuous, $q_j(t)$ is well-defined.

Theorem 2 (Ogier [16]). *Flow x^* , defined by*

$$x_{ij}^*(t) = \begin{cases} x_{ij}^\tau(t), & \text{if } q_i(t) = q_j(t) = \tau \\ u_{ij}(t), & \text{if } q_i(t) > q_j(t) \\ -u_{ij}(t), & \text{if } q_i(t) < q_j(t) \end{cases}$$

is a τ -maximum flow for all $\tau \in (0, T]$.

The proof of this theorem is technical. The intuition is that if, for $t \in (\theta - 1, \theta]$, there is a σ -maximum flow with $\sigma \geq t$ such that node $i(\theta)$ is on the source side of the cut defined by this flow, and $j(\theta)$ is on the sink side, then the flow on the arc (i, j) should be at capacity in this interval. If $i(\theta)$ and $j(\theta)$ are always on the same side of the cut corresponding to any τ -maximum flow, then the flow on the arc at time t should be determined by the τ -maximum flow with the latest completion time, $x^{q_j(\theta)}$, if this time $q_j(\theta)$ is after t ; and if not, then the flow should be determined by the maximum flow on $\mathcal{N}_\theta$, which is x^0 .

If W is known, then x^0 can be computed with $|W|$ maximum flows in a network the same size as $\mathcal{N}$, and x^* can be computed with an additional $|W|$ maximum flows in each H_τ , for all $\tau \in W$. Ogier describes a method to compute W in a piecewise fashion, computing $W \cap (\theta, \theta + 1]$ for $\theta = 0, \dots, k - 1$. Define the minimum cut function $\bar{\kappa}$ by setting $\bar{\kappa}(\tau)$ equal to the value of the minimum cut in H_τ .

Lemma 2. *Within the domain $(\theta, \theta + 1]$ for $\theta \in \{0, \dots, k - 1\}$, $\bar{\kappa}$ is concave and piecewise linear with at most $n - 2$ breakpoints.*

Proof. For $\tau \in (\theta, \theta + 1]$, $\theta \in \{0, \dots, k - 1\}$, $\bar{\kappa}(\tau)$ is the minimum function of a set of linear functions with positive slope: all capacities are linear, increasing functions of τ on the unit interval; hence, the value of all cuts are also linear, increasing functions of τ . This implies that $\bar{\kappa}$ is concave. Since R_τ is increasing in τ , changes in the slope of the cut function imply an increase in $|R_\tau \cap \mathcal{N}_{\theta+1}|$. This together with the fact that $d_{\theta+1} \in R_\tau$ and $s_{\theta+1} \notin R_\tau$ implies that $\bar{\kappa}(\tau)$ has at most $n - 2$ breakpoints in a unit interval.

Ogier [16] computes each subset $W \cap (\theta, \theta + 1]$ by finding τ such that $R_{\theta+}(\tau) = R_{\theta+1}(\tau)$, computing R_τ , and recursing on the subintervals $(\theta, \tau]$ and $(\tau, \theta + 1]$ until the interval does not properly contain elements of W . Since $\bar{\kappa}$ is concave in the interval $(\theta, \theta + 1]$, the τ that satisfies $R_{\theta+}(\tau) = R_{\theta+1}(\tau)$ lies in the interval. Lemma 2 implies that $|W|$ is at most nk . Ogier uses one maximum flow computation on a graph with at most nk vertices at each recursive step, yielding an $O(n^4 k^4)$ algorithm to compute W . Substituting the fastest strongly polynomial maximum flow algorithm — the preflow-push algorithm of Goldberg and Tarjan [9], Ogier's approach leads to an $O(k^3 m n^2 \log(kn^2/m))$ algorithm to compute x^* .

1.3 Parametric Maximum Flows

Our main contribution is to generalize the parametric maximum flow algorithm of Gallo, Grigoriadis, and Tarjan [8] to speed up the computation of W and

the x^τ , $\tau \in W$, as defined in the previous section. In this section, we review the parametric maximum flow algorithm and the preflow-push maximum flow algorithm on which it is based.

Gallo, Grigoriadis, and Tarjan [8] present several algorithms based on a *parametric preflow algorithm*. Given a graph G on n vertices and m arcs such that capacities of arcs leaving the source are nondecreasing functions of θ , capacities of arcs entering the sink are nonincreasing functions of θ , and all other capacities are constant, the parametric preflow algorithm finds a maximum flow and a minimum cut for each value of θ in an increasing sequence $\theta_1 < \dots < \theta_k$. The sequence may be computed on-line.

The parametric preflow algorithm is a parameterized version of the Goldberg and Tarjan [9] preflow-push algorithm for computing maximum flows. The preflow-push algorithm maintains at all times a feasible preflow and a valid labeling. A *feasible preflow* is a flow f satisfying arc capacity constraints $f_{ij} \leq u_{ij}$ and relaxed node conservation constraints $\sum_{i \in V} f_{ij} \geq 0$ for all $j \in V - \{s\}$. The quantity $\sum_{i \in V} f_{ij}$, when strictly positive, is called the *excess* at node j . A *valid labeling* l for preflow f is a function from the vertices to the nonnegative integers satisfying $l(s) = n$, $l(d) = 0$, $l(i) \geq 0$, and $l(i) \leq l(j) + 1$ for all (i, j) such that $f_{ij} < u_{ij}$.

The preflow-push algorithm typically starts with the preflow defined by saturating all arcs leaving the source, and the labeling $l(i) = 0$ for all $i \neq s$. Throughout the course of the algorithm, labels of nodes may increase but never decrease. The algorithm terminates when no nodes besides the source or sink contain excess. The entire analysis of the algorithm depends on the fact that the labels can only increase, and that at all times there is a path of residual arcs from any vertex to either the source or the sink, implying that no node has label greater than $2n$. The bound on the run time of the algorithm is determined by bounding the number of times each node is relabeled.

The analysis of this algorithm remains valid as long as it starts with any feasible preflow and corresponding valid labeling. There is nothing that requires the algorithm to start with the stated preflow and labeling. This fact is exploited by Gallo, et al. in their generalization of this algorithm to solve the parametric maximum flow problem.

The parametric preflow algorithm starts with capacities determined by the smallest value of the parameter θ , i.e. θ_1 , and computes a maximum flow. Arc capacities are then increased to the next largest value of θ . If there are arcs from s to nodes with label less than $l(s)$, the flow on these arcs is increased to meet the new capacities. The flow on arcs entering the sink is decreased, if necessary, to meet the new capacities, increasing the excess at the nodes adjacent to the sink. This results in a new, feasible preflow for which the previous labeling is still valid, since all saturated arcs in the previous preflow remain saturated in the new preflow. Thus the preflow-push algorithm will compute a maximum flow in this network with updated capacities.

The same bound on the run time of the preflow-push algorithm is also a bound on the run time of the parametric preflow push algorithm: $O(nm \log(n^2/m))$

time [8,9] (assuming the number of values of θ is not more than $m \log(n^2/m)$). This is because the bound on the run-time of the preflow-push algorithm depends only on the number of times a node is relabeled, node labels never decrease in either algorithm, and all node labels are $\leq 2n$.

Using this parametric preflow algorithm, Gallo, Grigoriadis, and Tarjan [8] describe an algorithm that finds all breakpoints of $\kappa(\theta)$, the minimum cut function of graph G , in the same asymptotic time. Like Ogier's algorithm, this algorithm relies on the concavity of κ . Once the breakpoints are found, the parametric preflow algorithm can be invoked again to compute the maximum flows and minimum cuts corresponding to these breakpoints.

2 Solving UMFP: A Generalized Parametric Maximum Flow Algorithm

In this section, we discuss the main contribution of this paper which is a generalization of the parametric maximum flow algorithm and the breakpoint algorithm of Gallo, Grigoriadis, and Tarjan [8] to solve the universally maximum flow problem with piecewise constant capacities. Our generalization of [8] enables us to reduce the time needed to compute the set of breakpoints of the universally maximum dynamic flow W , and the τ -maximum flows x^τ , $\tau \in W$ that are necessary to compute the optimal flow x^* as detailed in Section 1.2. We require $O(k^2nm \log(kn^2/m))$ time to do this. Since x^0 can be computed in $O(knm \log(n^2/m))$ time using k calls to Goldberg and Tarjan's push-relabel maximum flow algorithm [9], this implies that the universally maximum dynamic flow x^* can also be computed in $O(k^2nm \log(kn^2/m))$ time, which improves the algorithm of Ogier [16] by a factor of $O(kn)$.

Our algorithm integrates the work of Gallo, et al. [8] into the framework of the Ogier algorithm. In Step 1, we use the parametric preflow algorithm of Gallo, et al. to compute the minimum cuts R_θ and θ -maximum flows x^θ in H_θ for $\theta = 1, \dots, k$. In Step 2, we generalize the breakpoint algorithm of Gallo, et al. to compute the minimum cuts R_τ and corresponding maximum flows x^τ for all $\tau \in (\theta - 1, \theta] \cap W$, for each $\theta = 1 \dots, k$.

We consider a parametric flow problem based on the graphs H_τ . Instead of considering the graphs H_τ for $\tau \in (0, k]$ as separate graphs, we consider one graph H on the same vertex set but with parameterized capacities, so that the capacities of arcs of H at time τ equal the capacities of H_τ . That is, $H(\tau) = H_\tau$. More precisely, using $\mathcal{N}_\theta$ to denote the θ^{th} copy of network $\mathcal{N}$ in H , an arc in $\mathcal{N}_\theta$ has capacity 0 for $0 \leq t \leq \theta - 1$, capacity $(t - \theta + 1)u_e(\theta)$ for $\theta - 1 < t \leq \theta$, and capacity $u_e(\theta)$ for $\theta < t \leq k$. An arc from $\mathcal{N}_\theta$ to $\mathcal{N}_{\theta+1}$ has capacity 0 for $0 \leq t \leq \theta$ and capacity $u_j(\theta)$ for $\theta < t \leq k$.

For the correctness and speed of their algorithm, Gallo et al. require that all arcs with parameterized capacities either leave the source or enter the sink, and this does not hold for H . However, the Gallo et al. requirements as stated in Section 1.3 are merely sufficient conditions. The following conditions are also sufficient, but more general [8].

1. After increasing θ , it must be possible to adjust the flow so that it is a feasible preflow and modify the labeling so that it remains valid without decreasing any labels.
2. The minimum cut function $\kappa(\tau)$ must be concave.

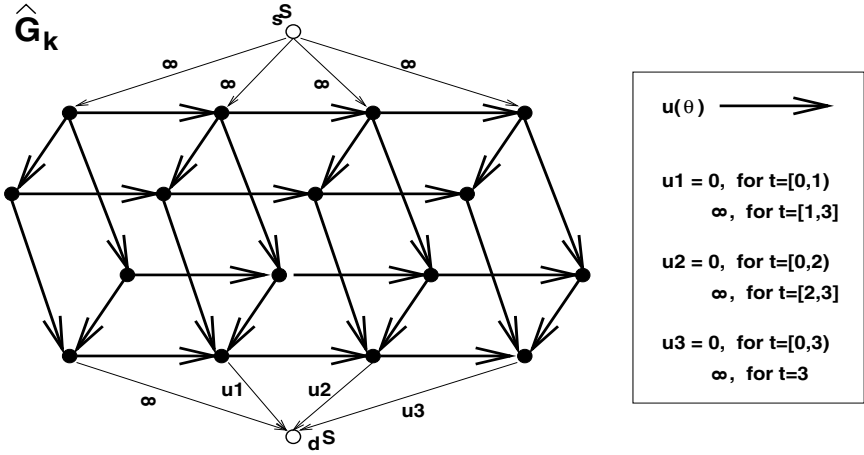
The second item is necessary for us to find the breakpoints W efficiently using a modified version of the Goldberg and Tarjan maximum flow algorithm and holds true here by Lemma 2. The first item is necessary to compute all corresponding maximum flows and minimum cuts in the same asymptotic time as one — this is a requirement of the parametric preflow algorithm as discussed in Section 1.3. We establish below (in Step 2) how to satisfy the first condition.

The two steps of our algorithm are detailed below. Figure 4 briefly summarizes the algorithm.

Step 1: Computing R_θ and x^θ for $\theta \in \{0, 1, \dots, k\}$. To compute the θ -maximum flows x^θ , $\theta \in \{0, \dots, k\}$, we construct a parametric flow problem on a modified G_k ($= H_k$). Recall that s_i is the copy of the source in $\mathcal{N}_i$, and d_i is the corresponding sink. We introduce a super source s^S with infinite capacity arcs (s^S, s_i) for each $i \in \{0, \dots, k\}$, a super sink d^S with arcs (d_i, d^S) for each $i \in \{0, \dots, k\}$ with capacity function that is infinite when $i \leq \theta$ and zero otherwise, and call this parametric network $\hat{G}_k$. (See Figure 3.) We then solve the parametric flow problem in $\hat{G}_k$ to find the maximum flows corresponding to each parameter $\theta \in \{0, 1, \dots, k\}$. These maximum flows are the flows x^θ for $\theta \in \{0, 1, \dots, k\}$. This is because, by Corollary 1, they correspond to cuts in $H(\theta) \equiv H_\theta$ of the same value; and, as a cut in H_θ , they keep this value even when all sinks d_i that are not in the original sink side of the cut are moved to the sink side: If d_i is not in the sink side of the cut for parameter θ in $\hat{G}_k$, the capacity of (d_i, d^S) for parameter θ must be finite, and is therefore zero. Thus the capacity of all arcs entering d_i in H_θ is also zero, and d_i can be moved to the sink side of the cut.

To solve this parametric flow problem, we reverse $\hat{G}_k$ by reversing the direction of every arc in $\hat{G}_k$, and apply the algorithm in [8] to this network. This parametric flow problem is of the form that is solved in [8]: all arcs that vary with θ are leaving the source, and the capacities are all increasing functions of θ . Thus, x^θ and R_θ for all $\theta \in \{0, \dots, k\}$ can be computed in the same asymptotic time as one maximum flow computation in the graph $\hat{G}_k$: $O(k^2 nm \log(kn^2/m))$ time.

Step 2: Computing R_τ and x^τ for all $\tau \in W$. To find the elements of W , and the corresponding τ -maximum flows within interval $(\theta - 1, \theta]$, we generalize the version of the parametric maximum flow algorithm [8] that finds all breakpoints of the minimum cut function $\bar{\kappa}(\theta)$. We start with $R_{\theta-1}$ and R_θ . The corresponding graphs are $H(\theta - 1)$ and $H(\theta)$. As τ increases from $\theta - 1$ to θ , the capacities of the arcs in $H \cap \mathcal{N}_\theta$ increase linearly from 0 to $u_E(\theta)$. Because the change in the capacity of the arcs is linear, the change in the minimum cut function is piecewise linear. By Lemma 2, this minimum cut function is also concave; thus it remains to show how to satisfy Condition 1 of [8].

Fig. 3. $\hat{G}_k$ for $k = 3$.

Step 1: Input: $G_k = (V_k, E_k)$

Output: θ -maximum flows x^θ and cuts R_θ for $\theta = 1, 2, \dots, k$.

$\hat{V} = V_k \cup \{s^S, d^S\}$.

$\hat{E} = E_k \cup \{(s^S, s_i) | i = 0, 1, \dots, k\} \cup \{(d_i, d^S) | i = 0, 1, \dots, k\}$.

with $\hat{u}_e(\theta) = u_e(\theta)$ for $e \in E$,

$\hat{u}_e(\theta) = \infty$ for e adjacent to s^S ,

$\hat{u}_e(\theta) = \infty$ for $e = (d_i, d^S)$ when $i \leq \theta$,

$\hat{u}_e(\theta) = 0$ for $e = (d_i, d^S)$ when $i > \theta$.

$\hat{G} = (\hat{V}, \hat{E})$ with capacity function $\hat{u}$.

$\hat{G}^R$ = the reverse network of $\hat{G}$.

$(x^1, x^2, \dots, x^k, R_1, R_2, \dots, R_k) = \mathbf{GGT}(\hat{G}^R, \theta = 1, 2, \dots, k)$.

Step 2: Input: $R_0 = \{d_i\}_{i=0}^k, R_1, R_2, \dots, R_k$.

Output: W and all τ -maximum flows x^τ for $\tau \in W$.

For $\theta = 1, 2, \dots, k$,

Modified $\mathbf{GGT}(R_\theta - R_{\theta-1})$ returns $W \cap (\theta - 1, \theta]$ and $R_{\tau_{i+1}} - R_{\tau_i}$

for all $\tau_i \in W \cap (\theta - 1, \theta]$, $\tau_i < \tau_{i+1}$.

For $\tau_1 < \tau_2 < \dots < \tau_n \in W \cap (\theta - 1, \theta]$,

$x_i^\tau|_{R_{\tau_{i+1}} - R_{\tau_i}} = \text{max flow on } R_{\tau_{i+1}} - R_{\tau_i} \cup \{v_1 = \overline{R}_{\tau_{i+1}}, v_2 = R_{\tau_i}\}$

Fig. 4. The two step algorithm to compute the τ -maximum flows needed for constructing a universally maximum flow.

Modified GGT. We run an iteration of the breakpoint algorithm in the reverse network H^R of H , restricted to the time interval $(\theta - 1, \theta]$ for some $\theta \in \{1, 2, \dots, k\}$. After computing the first maximum flow, we increase the current value of θ by ϵ , thus increasing the capacity of arcs in $\mathcal{N}_\theta$ by ϵ . (Here, ϵ is determined by the breakpoint algorithm in [8].) To adjust the preflow, we then increase the current flow in $\mathcal{N}_\theta$ by $\epsilon \frac{x_{ij}(\theta)}{u_{ij}(\theta)}$ for each arc $(i(\theta), j(\theta))$. Recall that $x_{ij}(\theta)$ equals the flow on arc (i, j) in $\mathcal{N}_\theta$. Since there are no arcs with positive capacity that enter $\mathcal{N}_\theta$ in H^R , there is no node in $\mathcal{N}_\theta$ that receives flow from outside $\mathcal{N}_\theta$. Hence this adjustment of the preflow maintains non-negative excess within $\mathcal{N}_\theta$, and the adjusted preflow is feasible. Since this update keeps previously saturated arcs saturated and empty arcs empty, the current labeling is still valid, and we have satisfied Condition 1.

The breakpoint algorithm is repeated for each unit interval of the form $(\theta - 1, \theta]$, $\theta = 1, \dots, k$ in $[0, k]$. This could take k parametric maximum flow computations in H^R , i.e. k times $O(k^2 nm \log(kn^2/m))$ time. To speed this up, we use the fact that we have already computed R_θ for all $\theta \in \{1, \dots, k\}$ in Step 1, and that, by Lemma 1, $R_{\theta-1} \subseteq R_\tau \subseteq R_\theta$, for all $\theta - 1 \leq \tau \leq \theta$. Thus, in searching for breakpoints of $\kappa(\tau)$ in the interval $(\theta - 1, \theta]$, we can restrict the computations performed for each unit interval to the subgraph of H^R on the vertex set $(R_\theta - R_{\theta-1}) \cup \{v_1, v_2\}$, where v_1 is the vertex resulting from the contraction of all vertices in $\overline{R_\theta}$ and v_2 is the vertex resulting from the contraction of $R_{\theta-1}$. Along with the breakpoints $\tau \in W \cap (\theta - 1, \theta]$ the algorithm also returns the R_τ implicitly as $R_\tau - R_{\theta-1}$.

We also need to compute the τ -maximum flows x^τ for all $\tau \in W$. We don't actually need to know the flow on every arc at every time step for each x^τ . By Theorem 2, it is sufficient to determine the values $x_{ij}^\tau(t)$ for t and τ such that $q_i(t) = q_j(t) = \tau$. Since x^τ is constant on unit intervals by definition (see Section 1.2), the definition of $q_j(t)$ in (2) implies that it is necessary to compute $x_{ij}^\tau(t)$ only for $t \leq \tau$ where $i(\lceil t \rceil), j(\lceil t \rceil) \in \overline{R_\tau}$ and $i(\lceil t \rceil), j(\lceil t \rceil) \in R_\sigma = R_{\tau+}$. By Corollary 2, this flow corresponds to a maximum flow in H_τ , and the particular flow amounts we require are on arcs contained in the subgraph of H^R induced by the vertices in $R_\sigma - R_\tau$. In order to compute x^τ restricted to this space-network interval, we must also specify the flow on all arcs entering and leaving this interval, i.e. all arcs adjacent but not contained in R_σ and R_τ . The flow on such arcs that are adjacent to R_τ in the relevant time interval is determined by definition of R_τ : all arcs entering R_τ must be saturated in x^τ , and all arcs leaving R_τ must be empty in x^τ . The flow on the remainder of these arcs, i.e. those adjacent to $\overline{R_\sigma}$, is similarly determined by citing one additional lemma of Ogier's.

Lemma 3 (Ogier [16]). *For each $\tau \in (0, T)$, $R_{\tau+}$ is a τ -minimum cut.*

In particular, for consecutive breakpoints $\tau < \sigma$ in W , $R_{\tau+} = R_\sigma$, so R_σ is also a τ -minimum cut. Thus the above observations for arcs entering and leaving R_τ also hold for R_σ , and now we can compute x^τ on H^R restricted to $R_\sigma \setminus R_\tau$.

Theorem 3. *A universally maximum flow with piecewise-constant capacities can be constructed in $O(k^2 mn \log(kn^2/m))$ time.*

Proof. We have shown above that Step 1 takes $O(k^2 mn \log(kn^2/m))$ time. To bound the run time of Step 2, we note that performing maximum flow computations on k graphs that together contain a total of nk vertices and $mk + nk$ arcs is bounded by the time to compute a maximum flow on a single graph with nk vertices and $(m + n)k$ arcs. Thus the total run time is $O(k^2 nm \log(kn^2/m))$.

Acknowledgments

I am grateful to Éva Tardos and Kevin Wayne for providing helpful comments on earlier drafts of this paper.

References

1. E. J. Anderson and P. Nash. *Linear Programming in Infinite-Dimensional Spaces*. John Wiley & Sons, 1987.
2. E. J. Anderson, P. Nash, and A. B. Philpott. A class of continuous network flow problems. *Mathematics of Operations Research*, 7:501–14, 1982.
3. E. J. Anderson and A. B. Philpott. A continuous-time network simplex algorithm. *Networks*, 19:395–425, 1989.
4. J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20:1–66, 1989.
5. R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR Methods and Models of Operations Research*, 37(1):31–58, 1993.
6. L. Fleischer. Faster algorithms for the quickest transshipment problem with zero transit times. In *Proceedings of the Ninth Annual ACM/SIAM Symposium on Discrete Algorithms*, pages 147–156, 1998. Submitted to SIAM Journal on Optimization.
7. L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
8. G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55, 1989.
9. A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of ACM*, 35:921–940, 1988.
10. B. Hajek and R. G. Ogier. Optimal dynamic routing in communication networks with continuous traffic. *Networks*, 14:457–487, 1984.
11. B. Hoppe. *Efficient Dynamic Network Flow Algorithms*. PhD thesis, Cornell University, June 1995. Department of Computer Science Technical Report TR95-1524.
12. B. Hoppe and É. Tardos. Polynomial time algorithms for some evacuation problems. In *Proc. of 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 433–441, 1994.
13. B. Hoppe and É. Tardos. The quickest transshipment problem. In *Proc. of 6th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 512–521, 1995.
14. E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21:517–527, 1973.

15. F. H. Moss and A. Segall. An optimal control approach to dynamic routing in networks. *IEEE Transactions on Automatic Control*, 27(2):329–339, 1982.
16. R. G. Ogier. Minimum-delay routing in continuous-time dynamic networks with piecewise-constant capacities. *Networks*, 18:303–318, 1988.
17. J. B. Orlin. Minimum convex cost dynamic network flows. *Mathematics of Operations Research*, 9(2):190–207, 1984.
18. A. B. Philpott. Continuous-time flows in networks. *Mathematics of Operations Research*, 15(4):640–661, November 1990.
19. A. B. Philpott and M. Craddock. An adaptive discretization method for a class of continuous network programs. *Networks*, 26:1–11, 1995.
20. W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science: Networks*. Elsevier Science Publishers B. V., 1995.
21. M. C. Pullan. An algorithm for a class of continuous linear programs. *SIAM J. Control and Optimization*, 31(6):1558–1577, November 1993.
22. M. C. Pullan. A study of general dynamic network programs with arc time-delays. *SIAM Journal on Optimization*, 7:889–912, 1997.
23. G. I. Stassinopoulos and P. Konstantopoulos. Optimal congestion control in single destination networks. *IEEE transactions on communications*, 33(8):792–800, 1985.
24. W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19:1602–1612, 1971.

Critical Extreme Points of the 2-Edge Connected Spanning Subgraph Polytope

Jean Fonlupt¹ and Ali Ridha Mahjoub²

¹ Équipe de Combinatoire, UF9 921, Université Pierre et Marie Curie,
4 place Jussieu, 75252 Paris Cedex 05 France

fonlupt@ec.ibp.fr

² LIMOS, Université de Clermont II, Complexe Scientifique des Cézeaux,
63177 Aubière Cedex, France

mahjoub@ucfma.univ-bpclermont.fr

Abstract. In this paper we study the extreme points of the polytope $P(G)$, the linear relaxation of the 2-edge connected spanning subgraph polytope of a graph G . We introduce a partial ordering on the extreme points of $P(G)$ and give necessary conditions for a non-integer extreme point of $P(G)$ to be minimal with respect to that ordering. We show that, if $\bar{x}$ is a non-integer minimal extreme point of $P(G)$, then G and $\bar{x}$ can be reduced, by means of some reduction operations, to a graph G' and an extreme point $\bar{x}'$ of $P(G')$ where G' and $\bar{x}'$ satisfy some simple properties. As a consequence we obtain a characterization of the perfectly 2-edge connected graphs, the graphs for which the polytope $P(G)$ is integral.

Keywords: Polytope, cut, 2-edge connected graph, critical extreme point.

1 Introduction

A graph $G = (V, E)$ is called *2-edge connected* if for every pair of nodes (u, v) there are at least two edge-disjoint paths between u and v . Given a graph $G = (V, E)$ and a weight function w which associates to each edge e a weight $w(e)$, the *2-edge connected subgraph problem* (TECSP) consists of finding a 2-edge connected subgraph $H = (V, F)$ of G , spanning all the nodes of G and such that $\sum_{e \in F} w(e)$ is minimum. This problem arises in the design of reliable transportation and communication networks [23], [24]. It is NP-hard in general. It has been shown to be polynomial in series-parallel graphs [26] and Halin graphs [25].

Given a graph $G = (V, E)$ and an edge subset $F \subseteq E$, the $0-1$ vector x^F of $\Re^E$ such that $x^F(e) = 1$ if $e \in F$ and $x^F(e) = 0$ if $e \in E \setminus F$ is called the *incidence vector* of F . The convex hull of the incidence vectors of the edge sets of the connected spanning subgraphs of G , denoted by $\text{TECP}(G)$, is called the *2-edge connected spanning subgraph polytope* of G .

Let $G = (V, E)$ be a graph. Given $b : E \rightarrow \Re$ and F a subset of E , $b(F)$ will denote $\sum_{e \in F} b(e)$. For $W \subseteq V$ we let $\overline{W} = V \setminus W$. If $W \subset V$ is a node subset of G , then the set of edges that have only one node in W is called a *cut* and denoted

by $\delta(W)$. (Note that $\delta(W) = \delta(\overline{W})$). If $W = \{v\}$ for some node $v \in V$, then we write $\delta(v)$ for $\delta(\{v\})$, $\delta(v)$ will be called a *node cut*. An *edge cutset* $F \subseteq E$ of G is a set of edges such that $F = \delta(S)$ for some non-empty set $S \subset V$. Given two vectors $w, x \in \mathbb{R}^E$, we let $\langle w, x \rangle = \sum_{e \in E} w(e)x(e)$.

If x^F is the incidence vector of the edge set F of a 2-edge connected spanning subgraph of G , then x^F is a feasible solution of the polytope $P(G)$ defined by

$$P(G) = \begin{cases} x(e) \geq 0 & \forall e \in E, \\ x(e) \leq 1 & \forall e \in E, \\ x(\delta(S)) \geq 2 & \forall S \subset V, S \neq \emptyset. \end{cases} \quad \begin{matrix} (1.1) \\ (1.2) \\ (1.3) \end{matrix}$$

Conversely, any integer solution of $P(G)$ is the incidence vector of the edge set of a 2-connected subgraph of G . Inequalities (1.1) and (1.2) are called *trivial constraints* and constraints (1.3) are called *cut constraints*.

The polytope $P(G)$ is a relaxation of the $\text{TECP}(G)$. It is also a relaxation of the *subtour polytope* of the traveling salesman problem, the set of all the solutions of the system given by inequalities (1.1)-(1.3) together with the constraints $x(\delta(v)) = 2$ for all $v \in V$. Thus minimizing $\langle w, x \rangle$ over $P(G)$ may provide a good lower bound for both the TECSP and the traveling salesman problem (see [12] [21]).

Using network flows [8] [9], one can compute in polynomial time a minimum cut in a weighted undirected graph. Hence the separation problem for inequalities (1.3) (i.e. the problem that consists of finding whether a given solution $\bar{y} \in \mathbb{R}^E$ satisfies inequalities (1.3) and if not to find an inequality which is violated by $\bar{y}$) can be solved in polynomial time. This implies by the ellipsoid method [13] that the TECSP can be solved in polynomial time on graphs G for which $\text{TECP}(G) = P(G)$. Mahjoub [20] called these graphs *perfectly 2-edge connected graphs* (perfectly-TEC). Thus an interesting question would be to characterize these graphs. In [19], Mahjoub showed that series-parallel graphs are perfectly-TEC. In [20] he described sufficient conditions for a graph to be perfectly-TEC.

In [10], Fonlupt and Naddef studied the graphs for which the polyhedron given by the inequalities (1.1) and (1.3) is the convex hull of the tours of G . (Here a tour is a cycle going through each node at least once). This is in connection with the graphical traveling salesman problem. They gave a characterization of these graphs in terms of excluded minors. (A minor of a graph G is a graph obtained from G by deletions and contractions of edges). A natural question that may arise here is whether or not one can obtain a similar characterization for perfectly-TEC graphs. The answer to this question is, unfortunately, in the negative. If we add the constraints $x(e) \leq 1$ for all $e \in E$, the approach developed by Fonlupt and Naddef [10] would not be appropriate. In fact, consider a non perfectly-TEC graph G (for instance a complete graph on four nodes or more). Subdivide G by inserting a new node of degree 2 on each edge and let $G' = (V', E')$ be the resulting graph. Clearly, each edge e of G' belongs to a 2-edge cutset. So $x(e) = 1$ for all $e \in E'$ is the unique solution of $P(G')$ and hence G' is perfectly-TEC. However, the graph G , which is a minor of G' , is not.

In this paper we study the fractional extreme points of the polytope $P(G)$. We introduce an ordering on the extreme points of $P(G)$ and give necessary

condition for a non-intger extreme point of $P(G)$ to be minimal with respect to that ordering. We show that if $\bar{x}$ is a minimal non-integer extreme point of $P(G)$, then G and $\bar{x}$ can be reduced, by means of some reduction operations, to a graph G' and an extreme point $\bar{x}'$ of $P(G')$, where G' and $\bar{x}'$ verify some simple properties. As a consequence, we obtain a characterization of perfectly-TEC graphs.

The polytope $\text{TECP}(G)$ has seen much attention in the last few years. Grötschel and Monma [14] and Grötschel, Monma and Stoer [15] [16] [17] [18] consider a more general model related to the design of minimum survivable networks. They discuss polyhedral aspects of this model and devise cutting plane algorithms along with computational results are presented. A complete survey of that model can be found in Stoer [24]. In [3] [4] Chopra studies the minimum k -edge connected spanning subgraph problem when multiple copies of an edge may be used. In [2], Barahona and Mahjoub characterize the polytope $\text{TECP}(G)$ for the class of Halin graphs. Baïou and Mahjoub [1] characterize the Steiner 2-edge connected subgraph polytope for series-parallel graphs.

Related work can be found in [5] [6] [7]. In [5] Cornuéjols, Fonlupt and Naddef study the TECSP when multiple copies of an edge may be used. They showed that when the graph is series-parallel, the associated polyhedron is completely described by inequalities (1.1) and (1.3). In [6] [7] Coullard et al. study the polytope, the extreme points of which are the edge-sets of the Steiner 2-node connected subgraphs of G .

This paper is organized as follows. In the next section we give more notation and definitions and present some preliminary results. In Section 3 we study an ordering on the extreme points of $P(G)$. In Section 4 we discuss some structural properties of the extreme points of $P(G)$. In Section 5 we prove our main result.

2 Notation, Definitions, and Preliminary Results

We consider finite, undirected and loopless 2-edge connected graphs, which may have multiple edges. We denote a graph by $G = (V, E)$ where V is the *node set* and E is the *edge set* of G . If e is an edge with endnodes u and v , then we write $e = uv$. If u is the endnode of an edge e , we say that u (resp. e) is incident to e (resp. u). If $W \subseteq V$, we denote by $G(W)$ the subgraph of G induced by W and by $E(W)$ the set of edges having both endnodes in W .

Given a solution $x \in \mathbb{R}^E$ of $P(G)$ we denote by $E_0(x)$, $E_1(x)$, $E_f(x)$ the sets of edges e such that $x(e) = 0$, $x(e) = 1$, $0 < x(e) < 1$, respectively. A cut $\delta(S)$ is said to be *tight* for x if $x(\delta(S)) = 2$. A node $v \in V$ is called *tight* for x if the cut $\delta(v)$ is tight for x . We will denote by $\tau(x)$ the set of tight cuts for x . A cut $\delta(W)$ will be called *proper* if $|W| > 2$ and $|\overline{W}| > 2$. Given a polyhedron P , we denote by $\dim(P)$ the dimension of P .

Let $G = (V, E)$ be a graph. Two cuts $\delta(W_1)$ and $\delta(W_2)$ of G are said to be *crossing* if $W_1 \cap W_2 \neq \emptyset$, $W_1 \not\subseteq W_2$, $W_2 \not\subseteq W_1$ and $V \setminus (W_1 \cup W_2) \neq \emptyset$. A family of cuts $\{\delta(W_1), \dots, \delta(W_k)\}$ is said to be *laminar* if $\delta(W_1), \dots, \delta(W_k)$ are pairwise non-crossing.

If $x \in \mathfrak{R}^E$ is a solution of $P(G)$ and $\delta(W)$ is a cut tight for x , we let

$$\begin{aligned}\tau(W, x) &= \{\delta(S) \in \tau(x) \mid S \subset W\}, \\ \tau(\overline{W}, x) &= \{\delta(S) \in \tau(x) \mid S \subset \overline{W}\}.\end{aligned}$$

In [5] Cornuéjols, Fonlupt and Naddef showed that if x is an extreme point of $P(G)$ and $\delta(W)$ is a cut tight for x , then x can be characterized by a system of equations whose proper tight cuts do not cross $\delta(W)$.

Lemma 2.1. [5] *Let x be an extreme point of $P(G)$ and $\delta(W)$ a cut of G tight for x . Then x is the unique solution of the system*

$$(2.1) \quad \begin{cases} x(e) = 0 & \forall e \in E_0(x), \\ x(e) = 1 & \forall e \in E_1(x), \\ x(\delta(W)) = 2, \\ x(\delta(S)) = 2 & \forall S \in \tau(W, x), \\ x(\delta(S)) = 2 & \forall S \in \tau(\overline{W}, x). \end{cases}$$

This lemma implies that any tight cut that crosses $\delta(W)$ induces a constraint which is redundant with respect to system (2.1). Note that system (2.1) may contain redundant equalities.

If x is a solution of $\mathfrak{R}^F$ where $F \subseteq E$ and T is a subset of F , then x_T will denote the restriction of x on T . If $S \subset V$, we let $\Gamma(S) = E(S) \cup \delta(S)$. If $W_1, W_2 \subset V$, (W_1, W_2) will denote the set of edges between W_1 and W_2 .

Throughout this paper $\bar{x}$ will denote a non-integer extreme point of $P(G)$. All the structures (polytopes, affine subspaces, ...) that will be considered in the sequel will refer to $\bar{x}$. So in the rest of the paper we will write $\tau(W)$, $\tau(\overline{W})$ instead of $\tau(W, \bar{x})$, $\tau(\overline{W}, \bar{x})$.

Let $\delta(W)$ be a cut of G tight for $\bar{x}$. Let L_0 be the affine subspace of $\mathfrak{R}^{\Gamma(W)}$ given by the constraints

$$L_0 = \begin{cases} x(e) = 0 & \forall e \in E_0(\bar{x}) \cap \Gamma(W), \\ x(e) = 1 & \forall e \in E_1(\bar{x}) \cap \Gamma(W). \end{cases}$$

Let $P(W) \subseteq L_0$ be the polytope given by

$$P(W) = \{x \in L_0 \mid 0 \leq x(e) \leq 1 \quad \forall e \in \Gamma(W), \\ x(\delta(S)) \geq 2 \quad \forall S \subset W, S \neq \emptyset\}.$$

And let $L(W)$ be the affine subspace of L_0 defined as

$$L(W) = \{x \in L_0 \mid x(\delta(S)) = 2 \quad \forall S \in \tau(W)\}.$$

In a similar way we define $P(\overline{W})$ and $L(\overline{W})$. Let $L_p(W)$ (resp. $L_p(\overline{W})$) be the projection of $L(W)$ (resp. $L(\overline{W})$) onto $\mathfrak{R}^{\delta(W)}$. Let

$$\begin{aligned}H(W) &= \{x \in \mathfrak{R}^{\delta(W)} \mid x(\delta(W)) = 2\}, \\ H^+(W) &= \{x \in \mathfrak{R}^{\delta(W)} \mid x(\delta(W)) \geq 2\}.\end{aligned}$$

We have the following lemma which is nothing but a reformulation of Lemma 2.1.

Lemma 2.2.

- i) $L_p(W) \cap L_p(\overline{W}) \cap H(W) = \bar{x}_{\delta(W)}$.
- ii) The maps $L(W) \longrightarrow L_p(W)$ and $L(\overline{W}) \longrightarrow L_p(\overline{W})$ are bijections.

3 Critical Extreme Points of $P(G)$

In this section we introduce the concept of critical extreme points and give our main result.

Suppose that $\bar{x}$ is a non-integer extreme point of $P(G)$. Let $\bar{x}'$ be the solution obtained by replacing some (but at least one) non-integer components of $\bar{x}$ by 0 or 1 (and keeping all the other components of $\bar{x}$ unchanged). Assume that $\bar{x}'$ is feasible for $P(G)$. Then $\bar{x}'$ can be written as a convex combination of extreme points of $P(G)$. If $\bar{y}$ is such an extreme point, then we shall say that $\bar{x}$ *dominates* $\bar{y}$ and we shall write $\bar{x} \succ \bar{y}$. Note that if $\bar{x}'$ is itself an extreme point of $P(G)$, then $\bar{y} = \bar{x}'$. Also note that $E_f(\bar{y}) \subset E_f(\bar{x})$ and if $\bar{x}(e) = 0$ or 1 for some edge $e \in E$, then $\bar{y}(e) = \bar{x}(e)$. Moreover, if a cut $\delta(W)$ is tight for both $\bar{x}$ and $\bar{x}'$, then $\delta(W)$ is also tight for $\bar{y}$. It may however exist cuts which are tight for $\bar{x}$ (resp. $\bar{y}$) but not tight for $\bar{y}$ (resp. $\bar{x}$).

It is clear that the relation $\succ$ defines a partial ordering on the extreme points of $P(G)$. The minimal elements of this ordering (*i.e.* the extreme points x for which there is no extreme point y such that $x \succ y$) correspond to the integer extreme points of $P(G)$. In what follows we define, in a recursive way, a rank function on the set of extreme points of $P(G)$.

The minimal extreme points of $P(G)$ will be called extreme points of *rank* 0. An extreme point x of $P(G)$ will be said of *rank* k , for fixed k , if x dominates the extreme points of rank $\leq k - 1$ and if there exists at least one extreme point of $P(G)$ of rank $k - 1$.

In [20], Mahjoub introduced the following operations:

- θ_1 : Delete an edge.
- θ_2 : Contract an edge whose both endnodes are of degree two.
- θ_3 : Contract a node subset W such that $G(W)$ is 2-edge connected.

And he proved the following result:

Theorem 3.1. [20] *If $G = (V, E)$ is perfectly-TEC and G' is a graph obtained by repeated applications of the operations $\theta_1, \theta_2, \theta_3$, then G' is perfectly-TEC.*

Now we are going to consider somewhat similar operations but defined with respect to a given solution $\bar{x}$ of $P(G)$.

- θ'_1 : Delete an edge e with $\bar{x}(e) = 0$.
- θ'_2 : Contract an edge e whose one of its endnodes is of degree 2.
- θ'_3 : Contract a node subset W such that $G(W)$ is 2-edge connected and $\bar{x}(e) = 1$ for all $e \in E(W)$.

Starting from a graph $G = (V, E)$ and a solution $\bar{x}$ of $P(G)$ and applying operations $\theta'_1, \theta'_2, \theta'_3$ we obtain a reduced graph $G' = (V', E')$ and a solution $\bar{x}' \in P(G')$. The following lemmas establish the relation between $\bar{x}$ and $\bar{x}'$. The proof of the first lemma is omitted because it is similar to that of Theorem 3.1.

Lemma 3.2. $\bar{x}$ is an extreme point of $P(G)$ if and only if $\bar{x}'$ is an extreme point of $P(G')$.

Lemma 3.3. $\bar{x}$ is an extreme point of $P(G)$ of rank 1 if and only if $\bar{x}'$ is an extreme point of $P(G')$ of rank 1.

Proof. Suppose that $\bar{x}$ is an extreme point of $P(G)$ of rank 1. By Lemma 3.2, $\bar{x}'$ is an extreme point of $P(G')$. If $\bar{x}'$ is not of rank 1, there must exist a fractional extreme point $\bar{x}''$ of $P(G')$ that is dominated by $\bar{x}'$. Now let $\bar{y}$ be the solution such that

$$\bar{y}(e) = \begin{cases} \bar{x}''(e) & \text{if } e \in E', \\ 1 & \text{if } e \in E_1(\bar{x}) \setminus E', \\ 0 & \text{if } e \in E_0(\bar{x}) \setminus E'. \end{cases}$$

By Lemma 3.2, $\bar{y}$ is an extreme point of $P(G)$. Moreover, $\bar{x}$ dominates $\bar{y}$. As $\bar{y}$ is fractional, this is a contradiction. The converse can be proved similarly. $\square$

An extreme point of $P(G)$ will be called *critical* if it is of rank 1 and if none of the operations $\theta'_1, \theta'_2, \theta'_3$ can be applied for it. Thus by Lemma 3.3, the characterization of the extreme points of rank 1 reduces to that of the critical extreme points of $P(G)$. In the rest of the paper we restrict ourselves to this type of extreme points.

Definition 3.4. Given a graph $G = (V, E)$ and a solution $x \in P(G)$, the pair (G, x) will be called a *basic pair* if the following hold:

1. $V = V_1 \cup V_2$ with $V_1 \cap V_2 = \emptyset$,
 $E = E_1 \cup E_2$ with $E_1 \cap E_2 = \emptyset$,
 $G_1 = (V_1, E_1)$ is an odd cycle,
 $G_2 = (V_1 \cup V_2, E_2)$ is a forest whose set of pending nodes is V_1 ,
and such that all the nodes of V_2 have degree at least 3.
2. $x(e) = \frac{1}{2}$ for $e \in E_1$ and $x(e) = 1$ for $e \in E_2$.
3. For any proper cut $\delta(W)$ of G , $x(\delta(W)) > 2$.

Figure 1 shows some examples of basic pairs. Figure 1 (a) is a wheel, Figure 1 (b) shows an example where one component of the forest is an edge with two pending nodes in V_1 . In Figure 1 (c) the forest is not a star. If $(G, \bar{x})$ is a basic pair, then we will say that G is a basic graph and $\bar{x}$ is a basic point.

Lemma 3.5. If $(G, \bar{x})$ is a basic pair, then $\bar{x}$ is an extreme point of $P(G)$.

Proof. Easy. $\square$

We state now the main result of the paper:

Theorem 3.6. If $\bar{x}$ is a critical extreme point of $P(G)$, then $(G, \bar{x})$ is a basic pair.

Let Ω be the class of graphs G such that G can be obtained from a basic graph by replacing some edges of E_2 by paths of length two. Note that the graphs

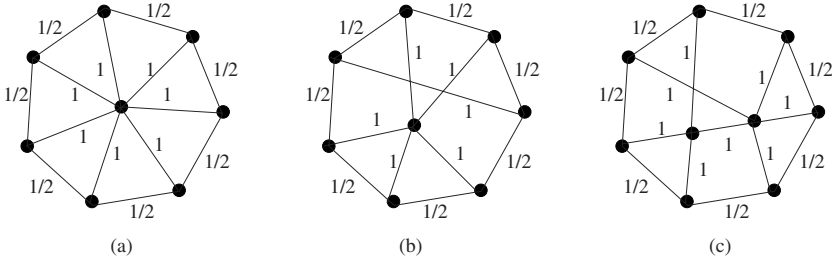


Fig. 1. Basic pairs

of Ω are not perfectly-TEC. In fact, let $\tilde{G} = (\tilde{V}, \tilde{E})$ and $\tilde{x} \in \mathcal{R}^{\tilde{E}}$ form a basic pair and $G = (V, E)$ a graph of Ω obtained from $\tilde{G}$ by inserting new nodes on some edges of the forest of $\tilde{G}$. Let $x \in \mathcal{R}^E$ be the solution of $P(G)$ such that $x(e) = \tilde{x}(e)$ if $e \in \tilde{E}$ and $x(e) = 1$ if $e \in E \setminus \tilde{E}$. As by Lemme 3.5, $\tilde{x}$ is an extreme point of $P(\tilde{G})$, x is a fractional extreme point of $P(G)$, and thus G is not perfectly-TEC.

A consequence of Theorem 3.6 is the following.

Corollary 3.7 *A graph G is perfectly-TEC if and only if G is not reducible to a graph of Ω by means of the operations $\theta_1, \theta_2, \theta_3$.*

Proof. Assume that $G = (V, E)$ reduces to a graph $G' = (V', E') \in \Omega$ by means of the operations $\theta_1, \theta_2, \theta_3$. By the remark above, G' is not perfectly-TEC. And thus by Theorem 3.1, G so is.

Conversely, if $\bar{x}$ is a fractional extreme point of $P(G)$, then there must exist an extreme point $\bar{y}$ of $P(G)$ of rank 1. By Lemma 3.3 together with Theorem 3.6, G and $\bar{y}$ can be reduced by operations $\theta'_1, \theta'_2, \theta'_3$ to a basic pair $(G', \bar{y}')$. If instead of applying θ'_2 we apply θ_2 , we obtain a graph of Ω . $\square$

The proof of Theorem 3.6 will be given in Section 5. In what follows we are going to establish some structural properties of the critical extreme points of $P(G)$. These properties will be useful in the sequel.

4 Structural Properties of Critical Extreme Points

Let $\bar{x}$ be a critical extreme point of $P(G)$. We have the following lemmas, for the proofs that are omitted, see [11].

Lemma 4.1.

- i) $\bar{x}(e) > 0$ for all $e \in E$.
- ii) G does not contain nodes of degree 2.
- iii) If $W \subseteq V$ is a node set such that $G(W)$ is 2-edge connected, then there exists an edge f in $E(W)$ with $0 < x(f) < 1$.
- iv) If $\delta(W)$ is a cut of G tight for $\bar{x}$, then both $G(W)$ and $G(\overline{W})$ are connected.

Lemma 4.2. *Every edge $f \in E_f(\bar{x})$ belongs to at least two tight cuts of system (2.1).*

Lemma 4.3. *Let $\delta(W)$ be a cut tight for $\bar{x}$ with $W = \{u, v\}$. Then*

- i) u and v are linked by exactly one edge, say f ,*
- ii) u and v are tight and $\bar{x}(f) = 1$,*
- iii) $\delta(W) \subseteq E_f(\bar{x})$.*

Remark 4.4. The converse of Lemma 4.3 also holds. That is, if $u, v \in V$ are two nodes satisfying *i)* and *ii)* of Lemma 4.3, then the cut $\delta(W)$ is tight for $\bar{x}$, where $W = \{u, v\}$. This implies that the constraint $\bar{x}(\delta(W)) = 2$ is redundant with respect to the equations $\bar{x}(\delta(u)) = 2$, $\bar{x}(\delta(v)) = 2$ and $\bar{x}(f) = 1$.

Lemma 4.5. *Let $\delta(W)$ be a proper cut tight for $\bar{x}$. If $G(W)$ and $G(\overline{W})$ are 2-edge connected, then for every two edges $f, g \in \delta(W)$, $\bar{x}(f) + \bar{x}(g) \leq 1$.*

Lemma 4.6. *Let $\delta(W)$ be a cut tight for $\bar{x}$. Then*

- i) $\delta(W)$ contains at least three edges,*
- ii) if $\delta(W) \cap E_1(\bar{x}) \neq \emptyset$, then either $\delta(W)$ or $\delta(\overline{W})$ is a node cut.*

Proof. *i)* If $\delta(W)$ is a 2-edge cutset, then we can show in a similar way as a result in [20] that either W or $\overline{W}$ is reduced to a single node. But this implies that G contains a node of degree 2, which contradicts Lemma 4.1 *ii)*.

ii) Suppose that $\delta(W)$ contains an edge $e_0 \in E_1(\bar{x})$. By Lemma 4.3 *iii)*, $|W| \neq 2 \neq |\overline{W}|$. So let us assume that $\delta(W)$ is a proper cut. We claim that $G(W)$ and $G(\overline{W})$ are both 2-edge connected. Indeed, since $\delta(W)$ is tight for $\bar{x}$, by Lemma 4.1 *iv)*, $G(W)$ and $G(\overline{W})$ are both connected. Now suppose, for instance, that $G(W)$ is not 2-edge connected. Then there exists a partition W_1, W_2 of W with $(W_1, W_2) = \{f\}$ for some $f \in E$. W.l.o.g. we may suppose that $e_0 \in (W_1, \overline{W})$. We have $\bar{x}(\delta(W)) = \bar{x}(\delta(W_1)) + \bar{x}(\delta(W_2)) - 2\bar{x}(f) = 2$. As $\bar{x}(\delta(W_1)) \geq 2$, $\bar{x}(\delta(W_2)) \geq 2$ and $\bar{x}(f) \leq 1$, it follows that $\bar{x}(f) = 1$ and $\delta(W_1)$ and $\delta(W_2)$ are both tight for $\bar{x}$. Hence $\delta(W_1) = \{e_0, f\}$, contradicting *i)*.

Consequently, both graphs $G(W)$ and $G(\overline{W})$ are 2-edge connected. Now let e_1 be an edge of $\delta(W) \setminus \{e_0\}$. As $\bar{x}(e_1) > 0$, we have that $\bar{x}(e_0) + \bar{x}(e_1) > 1$. Since $\delta(W)$ is proper, this contradicts Lemma 4.5. $\square$

Lemma 4.7. *If $\delta(W)$ is a proper cut tight for $\bar{x}$, then*

- i) $E(W) \cap E_f(\bar{x}) \neq \emptyset$, $E(\overline{W}) \cap E_f(\bar{x}) \neq \emptyset$,*
- ii) $G(W)$ and $G(\overline{W})$ are both 2-edge connected,*
- iii) $\bar{x}(e) < 1$ for all $e \in \delta(W)$,*
- iv) $|\delta(W)| \geq 4$,*
- v) $P(W) \cap H(W)$, $P(W) \cap H^+(W)$, $P(\overline{W}) \cap H(W)$ and $P(\overline{W}) \cap H^+(W)$ are integer polyhedra.*

Proof. *i)* Assume for instance that $E(W) \cap E_f(\bar{x}) = \emptyset$. Hence $G(W)$ cannot contain an induced 2-edge connected subgraph. For otherwise, G would be reducible by θ'_3 to a smaller graph, which contradicts the fact that $\bar{x}$ is critical. In consequence, by Lemma 4.1 *iv)*, it follows that $G(W)$ is a tree. Let u and v be two pendant nodes of this tree. As $\bar{x}(\delta(u)) \geq 2$, it follows that $\bar{x}(\delta(u) \cap \delta(W)) \geq 1$. Similarly we have $\bar{x}(\delta(v) \cap \delta(W)) \geq 1$. Since $\delta(W)$ is tight for $\bar{x}$, this implies that these inequalities are satisfied with equality and that $\delta(W) = (\delta(u) \cap \delta(W)) \cup (\delta(v) \cap \delta(W))$. Thus u and v are the only pendant nodes of $G(W)$ and, in consequence, $G(W)$ is a path P . Moreover, as G does

not contain nodes of degree 2, P must consist of a single edge. Thus $W = \{u, v\}$, which contradicts the fact that $\delta(W)$ is a proper cut.

ii) Suppose for instance that $G(W)$ is not 2-edge connected. As by Lemma 4.1 *iv)* $G(W)$ is connected, there exists a partition W_1, W_2 of W and an edge $f \in E(W)$ such that $(W_1, W_2) = \{f\}$. We have $\bar{x}(\delta(W)) = \bar{x}(\delta(W_1)) + \bar{x}(\delta(W_2)) - 2\bar{x}(f) = 2$. As $\bar{x}(\delta(W_1)) \geq 2$, $\bar{x}(\delta(W_2)) \geq 2$ and $\bar{x}(f) \leq 1$, it follows that $\bar{x}(f) = 1$ and $\delta(W_1)$ and $\delta(W_2)$ are both tight for $\bar{x}$. By Lemma 4.6 *ii)* this implies that $|W_1| = |W_2| = 1$. Hence $|W| = 2$, a contradiction

iii) If $\bar{x}(e') = 1$ for some $e' \in \delta(W)$, as $\bar{x}(e) > 0$ for all $e \in \delta(W)$, by *ii)* together with Lemma 4.5 it follows that $\bar{x}(e) = 0$ for all $e \in \delta(W) \setminus \{e'\}$. Thus $\bar{x}(\delta(W)) < 2$, which is impossible.

iv) This is a consequence of *ii)*, *iii)*, Lemma 4.5 and the fact that $\delta(W)$ is tight for $\bar{x}$.

v) Suppose, for instance, that $P(W) \cap H^+(W)$ is not integer and let $y \in \mathbb{R}^{\Gamma(W)}$ be a fractional extreme point of $P(W) \cap H^+(W)$. Let $x_0 \in \mathbb{R}^E$ be the solution such that

$$x_0(e) = \begin{cases} y(e) & \text{if } e \in \Gamma(W), \\ 1 & \text{if } e \in E(\bar{W}). \end{cases}$$

Since by *ii)* $G(\bar{W})$ is 2-edge connected, x_0 is a solution of $P(G)$. Moreover, as x_0 is the unique solution of the system given by the constraints defining y together with $x(e) = 1$, for all $e \in E(\bar{W})$, x_0 is an extreme point of $P(G)$. Furthermore, as $E(\bar{W}) \cap E_f(\bar{x}) \neq \emptyset$ we have that $\bar{x} \succ x_0$. As x_0 is fractional this contradicts the fact that $\bar{x}$ is critical.

Thus $P(W) \cap H^+(W)$ is integer. Since $P(W) \cap H(W)$ is a face of $P(W) \cap H^+(W)$, it is also integer. The proof for $P(\bar{W}) \cap H(W)$ and $P(\bar{W}) \cap H^+(W)$ is similar. $\square$

5 Proof of Theorem 3.6

In this section we shall prove our main result. To this end, we first establish the following result, its proof uses elements of convex analysis. For the basic definitions the reader is referred to Rockafellar [22].

Proposition 5.1. *If $\bar{x}$ is critical, then the set of proper tight cuts for $\bar{x}$ is empty.*

Proof. Assume the contrary and let $\delta(S)$ be a proper cut tight for $\bar{x}$. We may suppose that $|S| \leq |\bar{S}|$ and that $|S|$ is minimum, that is $|S| \leq \min(|S'|, |\bar{S}'|)$ for any proper cut $\delta(S')$ tight for $\bar{x}$. We have the following claims.

Claim 1.

- i) $\bar{x}(e) = \frac{1}{2}$ for all $e \in E_f(\bar{x})$,
- ii) $|\delta(Z)| = 4$ for every proper cut $\delta(Z)$ tight for $\bar{x}$.

Proof. See [11].

Now let $\delta(W)$ be a proper tight cut for $\bar{x}$. Then by Claim 1 there are four edges $\{e_1, \dots, e_4\}$ such that $\delta(W) = \{e_1, e_2, e_3, e_4\}$. As a consequence, the subspace $H(W) = \{x \in \mathbb{R}^{\delta(W)} \mid \sum_{i=1, \dots, 4} x(e_i) = 2\}$ contains six points with two components equal to 1 and two components equal to zero. Let $x^1, \dots, x^6$ be

those points. As $\bar{x}(e_i) = \frac{1}{2}$ for $i = 1, \dots, 4$, by a suitable ordering we can assume that $\bar{x}_{\delta(W)} = \frac{1}{2}(x^i + x^{i+3})$ for $i = 1, 2, 3$. Let us denote by L_i the line (subspace of dimension 1) generated by the points x^i, x^{i+3} for $i = 1, 2, 3$ and by Pl_i the plane (subspace of dimension 2) generated by the lines L_j, L_k ; $j \neq i \neq k$.

Claim 2.

- i) A line L_i , $i = 1, 2, 3$ cannot belong to both $L_p(W)$ and $L_p(\bar{W})$.
- ii) $L_p(W) \cap H(W)$ ($L_p(\bar{W}) \cap H(W)$) is either a line L_i or a plane Pl_i , for some $i \in \{1, 2, 3\}$.

Proof. i) As the lines L_i , $i = 1, 2, 3$ are contained in $H(W)$, the statement follows from Lemma 2.2 i).

ii) The polyhedron $P(W) \cap L(W) \cap H(W)$ is a face of the polyhedron $P(W) \cap H^+(W)$ and is therefore integer. Let $M(W)$ be the projection of $P(W) \cap L(W) \cap H(W)$ onto $\mathbb{R}^{\delta(W)}$. Hence $M(W)$ is integer and $M(W) \subseteq (L_p(W) \cap H(W))$. On the other hand, $P(W) \cap L(W) \cap H(W)$ is the smallest face of $P(W) \cap H^+(W)$ containing $\bar{x}_{\Gamma(W)}$. As $\bar{x}_{\delta(W)} \in M(W)$ and $\bar{x}_{\delta(W)}$ is fractional, $M(W)$ must contain at least two points x^i, x^{i+3} for some $i \in \{1, 2, 3\}$. (For otherwise $\bar{x}_{\delta(W)}$ could not be written as a convex combination of integer extreme points of $M(W)$). Thus $L_i \subset M(W)$ and hence $L_i \subset (L_p(W) \cap H(W))$. Similarly there is a line L_j , $j \neq i$, such that $L_j \subset (L_p(\bar{W}) \cap H(W))$.

Furthermore $M(W)$ cannot contain the six points $x^1, \dots, x^6$. For otherwise one would have $L_p(W) \cap H(W) = H(W)$, which implies, by Lemma 2.2 i), that $L_p(\bar{W}) \cap H(W) = \{\bar{x}_{\delta(W)}\}$. But this contradicts the fact that $L_j \subset (L_p(\bar{W}) \cap H(W))$. Thus $M(W)$ has either two or four points among $x^1, \dots, x^6$, implying that either $L_p(W) \cap H(W) = L_i$ or $L_p(W) \cap H(W) = Pl_i$.

Similarly the statement holds for $L_p(\bar{W}) \cap H(W)$. $\square$

Using Claim 2 we can show the following

Claim 3. If $\dim(L_p(W) \cap H(W)) = 2$, then $\delta(W)$ does not cross any tight cut for $\bar{x}$.

Proof. See [11].

By Claim 1, it follows that any cut $\delta(W)$ (proper or not) tight for $\bar{x}$ contains either three or four edges. Thus $\dim(L_p(W) \cap H(W)) \leq 3$. In what follows we will be mainly interested by the cases where the description of $L_p(W) \cap H(W)$ contains either one or two nontrivial equations.

If $L_p(W) \cap H(W)$ contains only one nontrivial equation say $\langle a, x \rangle = b$, then this constraint corresponds to the cut equation $x(\delta(W)) = 2$. If we consider the solution $y \in \mathbb{R}^{\Gamma(W)}$ such that $y(e) = \bar{x}(e) + \epsilon$, for $\epsilon > 0$, if e is a fractional edge of $\delta(W)$ and $y(e) = 1$ if not, then y is a solution of $P(W)$. Moreover the projection of y onto $\mathbb{R}^{\delta(W)}$, $y_{\delta(W)}$ verifies $\langle a, x \rangle > b$. In this case we will say that $\langle a, x \rangle = b$ is *relaxable*.

Now suppose that the description of $L_p(W) \cap H(W)$ is given by two nontrivial constraints

$$\begin{aligned} \langle a_1, x \rangle &= b_1, \\ \langle a_2, x \rangle &= b_2. \end{aligned}$$

Note that in this case $L_p(W) \cap H(W)$ is a plane.

A constraint $\langle a_i, x \rangle = b_i$, $i \in \{1, 2\}$ is said to be *relaxable* if there exists an affine subspace $L'(W) \subset \mathfrak{R}^{\Gamma(W)}$ containing $L(W)$ and a solution $y \in P(W) \cap L'(W)$ whose projection onto $\mathfrak{R}^{\delta(W)}$ verifies

$$\begin{aligned} \langle a_i, x \rangle &> b_i, \\ \langle a_j, x \rangle &= b_j. \end{aligned}$$

where $\{j\} = \{1, 2\} \setminus \{i\}$.

Now suppose that $\delta(W)$ is a proper cut tight for $\bar{x}$. Suppose that the cuts of $\tau(W)$ form a laminar family. A cut $\delta(T)$ of $\tau(W)$ is said to be *maximal* with respect to W if there is no a tight cut $\delta(Z)$ of $\tau(W)$ with $T \subset Z \subset W$. Let $\delta(W_1), \dots, \delta(W_r)$ be the tight cuts maximal with respect to W . Let

$$E' = \bigcup_{1 \leq i \leq r} \delta(W_i) \cup \delta(W).$$

In what follows we will denote by $L_q(W)$ the projection of $L(W)$ onto $\mathfrak{R}^{E'}$.

Claim 4. i) $L_p(W)$ is the projection of $L_q(W)$ onto $\mathfrak{R}^{\delta(W)}$.

ii) A solution $x \in \mathfrak{R}^{E'}$ belongs to $L_q(W)$ if and only if $x_{\delta(W_i)} \in L_p(W_i) \cap H(W_i)$ for $i = 1, \dots, r$.

Proof. See [11].

By Claim 4, if $L_q(W)$ is given by a system of the form

$$L_q(W) = \{x \in \mathfrak{R}^{E'} \mid \langle A_i, x \rangle = b_i; \ i \in I\},$$

where I is a set of indices, then there exists a partition $I_1, \dots, I_r$ of I such that

$$L_p(W_i) \cap H(W_i) = \{x \in \mathfrak{R}^{\Gamma(W_i)} \mid \langle A_j, x \rangle = b_j; \ j \in I_i\},$$

for $i = 1, \dots, r$. Let $\langle A_{i_0}, x \rangle = b_{i_0}$ be an equation with $i_0 \in I$. Suppose that $\langle A_{i_0}, x \rangle = b_{i_0}$ is relaxable. Consider the affine subspace

$$L'_q(W) = \{x \in \mathfrak{R}^{\Gamma(W)} \mid \langle A_i, x \rangle = b_i, \ i \in I \setminus \{i_0\}\}.$$

We have that $L_q(W) \subset L'_q(W)$. Thus there is an affine subspace $L'(W) \subset \mathfrak{R}^{\Gamma(W)}$ containing $L(W)$ whose projection onto $\mathfrak{R}^{E'}$ is $L'_q(W)$. Moreover there is a point $y \in P(W) \cap L'(W)$ whose projection onto $\mathfrak{R}^{E'}$ verifies

$$\begin{aligned} \langle A_{i_0}, y \rangle &> b_{i_0}, \\ \langle A_i, y \rangle &= b_i, \ i \in I \setminus \{i_0\}. \end{aligned}$$

We say that the relaxation of $\langle A_{i_0}, x \rangle = b_{i_0}$ is *valid* if at least one of such points satisfies the constraint $x(\delta(W)) \geq 2$.

Remark 5.2. If $L_q(W) \not\subset H(W)$, then y can be chosen so that $y(\delta(W)) = 2$, and thus the relaxation of $\langle A_{i_0}, x \rangle = b_{i_0}$ is valid.

Now suppose that $L_p(W) \cap H(W)$ is a plane. So w.l.o.g. we may suppose that $L_p(W) \cap H(W) = Pl_1$ and therefore, a description of $L_p(W) \cap H(W)$ is given by

system (5.3). The cut $\delta(W)$ is said to be *good* for W if $\dim(L_p(W) \cap H(W)) = 2$ and if in the description given by (5.3), at least one of the equations has a valid relaxation.

Now we turn to the crucial point in the proof.

Claim 5. All the proper cuts tight for $\bar{x}$ are good.

Proof. Suppose for instance that $\delta(W)$ is not good for W . We may suppose that $|W|$ is minimum, that is, all the proper tight cuts $\delta(Z)$ with $Z \subset W$ are good for Z . Consequently, by Claim 3, these cuts form a laminar family. Let $\delta(W_1), \dots, \delta(W_r)$ be the maximal tight cuts of W . Let E' and $L_q(W)$ be as defined in Claim 4. In what follows we are going to give a description of the affine subspace $L_q(W)$. For this, let us first note that, by Claim 1, either $|\delta(W_i)| = 3$ or $|\delta(W_i)| = 4$, for $i = 1, \dots, r$.

If $|W_i| = 1$, and $\delta(W_i) = \{e_1, e_2, e_3, e_4\}$, (resp. $\delta(W_i) = \{e_1, e_2, e_3\}$ with $e_3 \in E_1(\bar{x})$), then $\delta(W_i)$ produces in $L_q(W)$ the constraint:

$$x(e_1) + x(e_2) + x(e_3) + x(e_4) = 2, \quad (5.4)$$

$$\text{(resp. } x(e_1) + x(e_2) = 1). \quad (5.5)$$

Note here that $L_p(W_i) \cap H(W_i)$ is given by inequality (5.4) (resp. (5.5) together with $x(e_3) = 1$), and thus (5.4) (resp. (5.5)) is relaxable.

If $W_i = \{u, v\}$ for some nodes $u, v \in V$, by Lemma 4.3 *i)* we have that $uv \in E$, $\bar{x}(uv) = 1$ and $\delta(u)$ and $\delta(v)$ are both tight for $\bar{x}$. This implies that $\bar{x}(\delta(W_i)) = 2$ is redundant with respect to $\bar{x}(\delta(u)) = 2, \bar{x}(\delta(v)) = 2$ and $\bar{x}(uv) = 1$. And thus $\delta(W_i)$ produces two equations of type (5.5).

If $\delta(W_i)$ is a proper cut, then by our minimality assumption, $\delta(W_i)$ is a good cut, and by Claim 1 *ii)*, $|\delta(W_i)| = 4$. Moreover if, for instance, $\delta(W_i) = \{e_1, e_2, e_3, e_4\}$, we may suppose, w.l.o.g. that the plane $L_p(W_i) \cap H(W_i)$ is given by system (5.3), and that at least one of the constraints of that system has a valid relaxation.

Let $Ax = b$ be the system given by the constraints of type (5.3), (5.4), (5.5). Let k be the number of constraints of this system. Let $E^* = E' \cap E_f(\bar{x})$. Let $L_q^*(W)$ be the projection of $L(W)$ onto E^* . Note that the projection of $L_q^*(W)$ onto $\Re^{\delta(W)}$ is $L_p(W)$. Also note that the maps $L(W) \longrightarrow L_q(W) \longrightarrow L_q^*(W) \longrightarrow L_p(W)$ are bijections. By Claim 4 it follows that

$$L_q^*(W) = \{x \in \Re^{E^*}; Ax = b\}.$$

Now remark that an edge $e \in E^* \setminus \delta(W)$ belongs to at most two maximal tight cuts $\delta(W_i)$. And an edge e of $\delta(W)$ belongs to at most one maximal tight cut $\delta(W_i)$. Thus $x(e)$ appears in at most two rows (resp. one row) of $Ax = b$, if $e \in E^* \setminus \delta(W)$ (resp. $e \in \delta(W)$). In what follows we are going to define connected components of A in a similar way as for graphs. We say that two edges f and g of E^* (or the corresponding columns of A) are *connected* if

1) there exists a sequence $f_1, \dots, f_q$ (the f_i may be the same) with $f_1 = f$ and $f_q = g$ such that $x(f_i)$ and $x(f_{i+1})$ appear in a row j of $Ax = b$ for $i = 1, \dots, q-1$, and

2) the set $\{f_1, \dots, f_q\}$ is maximal with respect to 1).

Let E_1 be a connected component of the column set of A , and let I_1 be the set of indices $i \in \{1, \dots, k\}$ such that some $x(f)$ with $f \in E_1$ appears in the equation $\langle A_i, x \rangle = b_i$. (The submatrix $A_{I_1}^{E_1}$ is a block of A). A point $y \in \mathbb{R}^{E^*}$ is a solution of $Ax = b$ if and only if y_{E_1} is a solution of the system $A_{I_1}^{E_1} x = b_{I_1}$ and $y_{E^* \setminus E_1}$ is a solution of $A_{I_2}^{E_2} x = b_{I_2}$ where $E_2 = E^* \setminus E_1$ and $I_2 = \{1, \dots, k\} \setminus I_1$. We claim that $E_1 \cap \delta(W) \neq \emptyset$. Indeed, as by Lemma 4.7 v) $P(W) \cap H(W)$ is integer and therefore $L_q^*(W) \cap H(W)$ so is, the system $\{Ax = b, x(\delta(W)) = 2\}$ has an integer solution y , say. Suppose that $E_1 \cap \delta(W) = \emptyset$. Define $x' \in \mathbb{R}^{E'}$ as $x'_{E_1} = y_{E_1}$ and $x'_{E' \setminus E_1} = \bar{x}_{E' \setminus E_1}$. Clearly, $x' \in L_q(W)$. Since $\bar{x}_{E_1}$ is fractional, we have that $x' \neq \bar{x}_{E'}$. Moreover, x' and $\bar{x}_{E'}$ have the same projection $\bar{x}_{\delta(W)}$ onto $\mathbb{R}^{\delta(W)}$. But this contradicts the fact that the projection $L_q(W) \rightarrow L_p(W)$ is a bijection.

A block $A_{I_1}^{E_1}$ of A will be called *active* if the projection of the affine subspace $\{x \in \mathbb{R}^{E^*} \mid \langle A_i, x \rangle = b_i; i \in I_1\}$ onto $\mathbb{R}^{\delta(W)}$ is contained in an hyperplane of the form

$$H = \{x \in \mathbb{R}^{\delta(W)} \mid \sum_{i=1, \dots, 4} c(e_i)x(e_i) = \beta\}.$$

In other words, a block $A_{I_1}^{E_1}$ is active if an equation of the form $\sum_{i=1, \dots, 4} c(e_i)x(e_i) = \beta$ is redundant with respect to that block. In that case we will say that the hyperplane H is *produced* by $A_{I_1}^{E_1}$.

Since the equation $\sum_{i=1, \dots, 4} c(e_i)x(e_i) = \beta$ can be obtained as a linear combination of the equations $\langle A_i, x \rangle = b_i$, $i \in I_1$, there must exist a vector $u = (u_1, \dots, u_{I_1})$ such that $(\sum_{i=1, \dots, 4} c(e_i)x(e_i) - \beta) = \sum_{i \in I_1} u_i(\langle A_i, x \rangle - b_i)$. As $u \neq 0$, we may suppose, w.l.o.g. that $u_{i_0} = 1$ for some $i_0 \in I_1$. As $\sum_{i \in I_1} u_i A_i(f) = 0$ for every $f \in E_1 \setminus \delta(W)$, for every row $i \in I_1$ one should have either $u_i = +1$ or $u_i = -1$ in such a way that if an edge f of $E_1 \setminus \delta(W)$ appears in two lines i and j , then $u_i + u_j = 0$. Moreover, starting from $u_{i_0} = +1$, the coefficients u_i , $i \in I_1 \setminus \{i_0\}$ are determined in a unique manner. As $c(e_i) \neq 0$ for at least one edge $e_i \in \delta(W)$, this implies that the rows $i \in I_1$ are linearly independent and that at most one hyperplane may be produced by an active block.

Let $t = |E_1 \cap \delta(W)|$ and $L_1 = \{x \in \mathbb{R}^{E_1 \cup \delta(W)} \mid \langle A_i, x \rangle = b_i, i \in I_1\}$. As the map $L_q(W) \rightarrow L_p(W)$ is a bijection, it follows that the projection of L_1 onto $\mathbb{R}^{\delta(W)}$ so is. Since the projection of L_1 onto $\mathbb{R}^{\delta(W)}$ is an hyperplane, it follows that $\dim(L_1) = 3$. Also as $A_{I_1}^{E_1} x = b_{I_1}$ is a non-redundant system, it follows that $|E_1 \cup \delta(W)| = |I_1| + 3$. Hence

$$|E_1| = |I_1| + t - 1. \quad (5.6)$$

Let l be the number of constraints $i \in I_1$ of type (5.4). (Note that for this type of constraints we have $b_i = 2$.) We then have

$$\sum_{i \in I_1} \langle A_i, \bar{x} \rangle = |I_1| + l. \quad (5.7)$$

On the other hand, as $\bar{x}(e) = \frac{1}{2}$ for all $e \in E_f(\bar{x})$, and every column corresponding to an edge of $E_1 \setminus \delta(W)$ (resp. $E_1 \cap \delta(W)$) contains exactly two 1's (resp. one 1), we have

$$\sum_{i \in I_1} < A_i, \bar{x} > = |E_1| - t + \frac{t}{2} = |E_1| - \frac{t}{2}. \quad (5.8)$$

Combining (5.6)-(5.8) we get $l = \frac{t}{2} - 1$. Thus t is even, and therefore either $t = 2$ (and $l = 0$) or $t = 4$ (and $l = 1$). As $L_p(W)$ is the projection of $L_q(W)$ onto $\Re^{\delta(W)}$, each hyperplane in the description of $L_p(W)$ is produced by one active block. We distinguish three cases:

Case 1) There is only one active block $A_{I_1}^{E_1}$ with $|E_1 \cap \delta(W)| = 2$, and thus $\dim(L_p(W)) = 3$.

Case 2) There is only one active block $A_{I_1}^{E_1}$ with $|E_1 \cap \delta(W)| = 4$. We claim that in this case $E_1 = E^*$ and $I_1 = \{1, \dots, k\}$. In fact, if there is a further block $A_{I_2}^{E_2}$, then as $E_1 \cap \delta(W) \neq \emptyset \neq E_2 \cap \delta(W)$ there must exist an edge $g \in E_1 \cap E_2 \cap \delta(W)$, which is impossible. Note that here there is a constraint of type (5.4) ($l = 1$) and, as in Case 1), $\dim(L_p(W)) = 3$.

Case 3) There are two active blocks $A_{I_1}^{E_1}, A_{I_2}^{E_2}$ with $I_1 \cup I_2 = \{1, \dots, k\}$, $E_1 \cup E_2 = E^*$, $|E_1 \cap \delta(W)| = |E_2 \cap \delta(W)| = 2$ and $\dim(L_p(W)) = 2$. Note that in this case, there is no constraints of type (5.4) ($l = 0$).

If Case 1 holds, then $L_p(W) \cap H(W)$ is a plane, which, by Claim 2, may be supposed to be given by system (5.3). Thus there are two scalars $\lambda, \mu \in \Re$ such that

$$\begin{aligned} L_p(W) &= \{x \in R^{\delta(W)} \mid \lambda(x(e_1) + x(e_2) - 1) + \mu(x(e_3) + x(e_4) - 1) = 0\} \\ &= \{x \in R^{\delta(W)} \mid \sum_{i=1, \dots, 4} c(e_i)x(e_i) = \beta\}. \end{aligned}$$

As $|E_1 \cap \delta(W)| = 2$, two coefficients among $c(e_1), \dots, c(e_4)$ must be equal to zero. In consequence, we may assume, w.l.o.g., that $L_p(W) = \{x \in R^{\delta(W)} \mid x(e_1) + x(e_2) = 1\}$. As $L_p(W) \cap H(W) \subset L_p(W)$, there must exist a solution $y' \in L_p(W) \setminus H(W)$. Moreover, y' can be chosen so that $y'(e_3) + y'(e_4) > 1$ and $y'(e_1) + y'(e_2) = 1$. Hence y' is the projection of a point y of $P(W) \cap L'(W)$ onto $\Re^{\delta(W)}$ where $L'(W)$ is an affine subspace containing $L(W)$. As $y(\delta(W)) \geq 2$, the constraint $x(e_3) + x(e_4) = 1$ has a valid relaxation. But this implies that $\delta(W)$ is good, a contradiction.

Now let us examine the two remaining cases. For this, one can first show that $k+l$ is even. And as either $l = 0$ or $l = 1$, $Ax = b$ contains at least one constraint of type (5.5) or (5.3). So we may suppose, for instance, that the first constraint of $Ax = b$ is of the form $< A_1, x > = 1$ and is relaxable. Thus there exists a solution $x^* \in (P(W) \cap L'(W))$ where $L'(W)$ is an affine subspace containing $L(W)$ such that

$$\begin{aligned} &< A_1, x > > 1 \\ &< A_i, x > = b_i, \quad i = 2, \dots, k. \end{aligned}$$

If Case 2 holds, then $\dim(L_p(W)) = 3$ and thus $L_p(W) \not\subset H(W)$. For otherwise, we would have $L_p(W) = H(W)$ and, in consequence, $L_p(W) \cap H(W)$ is neither a line nor a plane, which contradicts Claim 2 ii).

Consequently, the equation $x(\delta(W)) = 2$ is non-redundant with respect to the system $Ax = b$. By Remark 5.2, x^* can be chosen so that $x^*(\delta(W)) = 2$. Moreover, as the polyhedron $P(W) \cap L(W) \cap H(W)$ is integer as well as its faces, there exists an integer solution $\bar{y} \in \mathbb{R}^{E^*}$ such that

$$\begin{aligned} & \langle A_1, \bar{y} \rangle > 1, \\ & \langle A_i, \bar{y} \rangle = b_i, \quad i = 2, \dots, k, \\ & \sum_{i=1, \dots, 4} \bar{y}(e_i) = 2. \end{aligned}$$

As the first constraint is of type (5.3) or (5.5), we have $\langle A_1, \bar{y} \rangle = 2$. Thus $\sum_{i=1, \dots, k} \langle A_i, \bar{y} \rangle = k + l + 1 = 2 \left(\sum_{f \in (E^* \setminus \delta(W))} \bar{y}(f) \right) + \sum_{e \in \delta(W)} \bar{y}(e)$. But the right hand side of the second equation is even, whereas $k + l + 1$ is odd, a contradiction.

Now consider Case 3. If $L_p(W) \not\subset H(W)$, then one can get a contradiction in a similar way as before. So suppose that $L_p(W) \subset H(W)$. As $\dim(L_p(W)) = 2$, $L_p(W)$ is a plane and thus may be supposed to be defined by system (5.3). Let $A_{I_1}^{E_1}$ be the block that produces the equation $x(e_1) + x(e_2) = 1$. Thus there is a vector u such that

$$\begin{aligned} \sum_{i \in I_1} u_i \langle A_i, x \rangle &= x(e_1) + x(e_2), \\ \sum_{i \in I_1} u_i b_i &= 1. \end{aligned}$$

Note that $b_i = 1$ for all $i \in I_1$ (since $l = 0$). Let I_1^+ (resp. I_1^-) be the set of rows for which the coefficient u_i is equal to $+1$ (resp. -1). We have $\sum_{i \in I_1} u_i = |I_1^+| - |I_1^-| = 1$. We can define in a similar way I_2^+ and I_2^- for the second block $A_{I_2}^{E_2}$. Thus $|I_1^+| + |I_2^+| = 2 + |I_1^-| + |I_2^-|$. Note that all the constraints of $Ax = b$ are of type (5.3) and (5.6) and thus the number of relaxable rows is greater than or equal to the number of rows which are not relaxable. Thus there exists an equation $\langle A_{i_0}, x \rangle = b_{i_0}$, $i_0 \in I_1^+ \cup I_2^+$ which is relaxable. W.l.o.g. we may assume that $i_0 \in I_2^+$ and $\langle A_{i_0}, x \rangle = b_{i_0}$ is the constraint $x(e_3) + x(e_4) = 1$. So there is a solution $y' \in \mathbb{R}^{E^*}$ such that

$$\begin{aligned} & \langle A_{i_0}, y' \rangle > b_{i_0}, \\ & \langle A_i, y' \rangle = b_i, \quad i = 1, \dots, k, \quad i \neq i_0. \end{aligned}$$

Hence $y'(e_1) + y'(e_2) = 1$ and $y'(e_3) + y'(e_4) > 1$. Moreover y' is the projection of a point of $P(W) \cap L'(W)$ where $L'(W)$ is an affine subspace containing $L(W)$. Since $y(\delta(W)) \geq 2$, this implies that $\delta(W)$ is a good cut. But this is a contradiction and our claim is proved. $\square$

Now as $\delta(S)$ is a proper cut tight for $\bar{x}$, By Claim 5, $\delta(S)$ is good for S and for $\bar{S}$. Thus $\dim(L_p(S) \cap H(S)) = 2$ and $\dim(L_p(\bar{S}) \cap H(S)) = 2$. In consequence, the affine space $L_p(S) \cap L_p(\bar{S}) \cap H(S)$ contains one of the three lines L_1, L_2, L_3 . But this contradicts Lemma 2.2 i), which finishes the proof of the proposition. $\square$

Since $\bar{x}$ is critical, $\bar{x}(e) > 0$ for all $e \in E$. Let $V_f(\bar{x})$ be the subset of nodes incident to at least one edge of $E_f(\bar{x})$. From Lemma 4.2 together with Proposition 5.1, it follows that every node of $V_f(\bar{x})$ is tight for $\bar{x}$. Let $G_f = (V_f(\bar{x}), E_f(\bar{x}))$. We claim that G_f does not contain pendant nodes. In fact, assume the contrary

and let v_0 be a pendant node of G_f . Let f_0 be the edge of G_f adjacent to v_0 . Since $\bar{x}(\delta(v_0)) \geq 2$, v_0 must be incident to at least two edges of $E_1(x)$. Since $\bar{x}(f_0) > 0$, this implies that v_0 is not tight, a contradiction. Moreover G_f cannot contain even cycles. In fact, if G_f contains an even cycle $(e_1, e_2, \dots, e_k)$, then the solution $\bar{x}' \in \mathcal{R}^E$ such that

$$\bar{x}'(e) = \begin{cases} \bar{x}(e) + \epsilon & \text{if } e = e_i, i = 1, 3, \dots, k-1, \\ \bar{x}(e) - \epsilon & \text{if } e = e_i, i = 2, 4, \dots, k, \\ \bar{x}(e) & \text{otherwise,} \end{cases}$$

where ϵ is a scalar sufficiently small, would be a solution of system (2.1) different from $\bar{x}$, which is impossible.

Thus G_f does not contain even cycles. Now, using b-matchings, it is not hard to see that the components of G_f consist of odd cycles. We claim that G_f consists of only one odd cycle. Indeed, suppose that G_f contains two odd cycles C^1 and C^2 . Consider the solution $\tilde{x}$ defined as

$$\tilde{x}(e) = \begin{cases} \frac{1}{2} & \text{if } e \in C^1, \\ 1 & \text{if } e \in E \setminus C^1. \end{cases}$$

Obviously, $\tilde{x} \in P(G)$. Moreover $\tilde{x}$ is an extreme point of $P(G)$ such that $\bar{x} \succ \tilde{x}$. But this contradicts the fact that $\bar{x}$ is critical.

Consequently, G_f consists of only one odd cycle, say C . Note that every node of C is adjacent to exactly one edge of $E_1(\bar{x})$. Also note that, as $\bar{x}$ is critical, the graph induced by $E_1(\bar{x})$ does not contain neither 2-edge connected subgraphs nor nodes of degree 2. Thus this graph is a forest without nodes of degree 2 and whose pendant nodes are the nodes of C . Hence G verifies condition 1 of Definition 3.4. Moreover it is easy to see that $\bar{x}$ is given by $\bar{x}(e) = \frac{1}{2}$ for all $e \in C$ and $\bar{x}(e) = 1$ for all $e \in E \setminus C$. As G does not contain proper tight cut we have that $\bar{x}$ also satisfies conditions 2 and 3 of Definition 3.4. Which implies that $(G, \bar{x})$ is a basic pair and our proof is complete.

References

1. Baïou, M., Mahjoub, A.R.: Steiner 2-edge connected subgraphs polytopes on series parallel graphs. *SIAM Journal on Discrete Mathematics* 10 (1997) 505-514
2. Barahona, F., Mahjoub, A.R.: On two-connected subgraphs polytopes. *Discrete Mathematics* 17 (1995) 19-34
3. Chopra, S.: Polyhedra of the equivalent subgraph problem and some edge connectivity problems. *SIAM Journal on Discrete Mathematics* 5 (1992) 321-337
4. Chopra, S.: The k -edge connected spanning subgraph polyhedron. *SIAM Journal on Discrete Mathematics* 7 (1994) 245-259
5. Coruégols, G., Fonlupt, J., Naddef, D.: The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming* 33 (1985) 1-27
6. Coullard, R., Rais, A., Rardin, R.L., Wagner, D.K.: The 2-connected Steiner subgraph polytope for series-parallel graphs. Report No. CC-91-23, School of Industrial Engineering, Purdue University (1991)

7. Coullard, R., Rais, A., Rardin, R.L., Wagner, D.K.: The dominant of the 2-connected Steiner subgraph polytope for W_4 -free graphs. *Discrete Applied Mathematics* 66 (1996) 33-43
8. Dinitz, E.A.: Algorithm for solution of a problem of maximum flow in a network unit with power estimation. *Soziet. mat. Dokl.* 11 (1970) 1277-1280
9. Edmonds, J., Karp, R.M.: Theoretical improvement in algorithm efficiency for network flow problems. *J. of Assoc. Comp. Mach.* 19 (1972) 254-264
10. Fonlupt, J., Naddef, D.: The traveling salesman problem in graphs with some excluded minors. *Mathematical Programming* 53 (1992) 147-172
11. Fonlupt, J., Mahjoub, A.R.: Critical extreme points of the 2-edge connected spanning subgraph polytope. Preprint (1998)
12. Goemans, M.X., Bertsimas, D.J.: Survivable Network, linear programming and the parsimonious property. *Mathematical Programming* 60 (1993) 145-166
13. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences combinatorial optimization. *Combinatorica* 1 (1981) 70-89
14. Grötschel, M., Monma, C.: Integer polyhedra arising from certain network design problem with connectivity constraints. *SIAM Journal of Discrete Mathematics* 3 (1990) 502-523
15. Grötschel, M., Monma, C., Stoer, M.: Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM Journal on Optimization* 2 (1992) 474-504
16. Grötschel, M., Monma, C., Stoer, M.: Polyhedral approaches to network survivability. In: Roberts, F., Hwang, F., Monma, C. (eds.): *Reliability of Computer and Communication Networks*, Vol. 5, Series in Discrete Mathematics and Computer Science AMS/ACM (1991) 121-141
17. Grötschel, M., Monma, C., Stoer, M.: Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research* 40/2 (1992) 309-330
18. Grötschel, M., Monma, C., Stoer, M.: Polyhedral and Computational Investigation for designing communication networks with high suivability requirements. *Operations Research* 43/6 (1995) 1012-1024
19. Mahjoub, A.R.: Two-edge connected spanning subgraphs and polyhedra. *Mathematical Programming* 64 (1994) 199-208
20. Mahjoub, A.R.: On perfectly 2-edge connected graphs. *Discrete Mathematics* 170 (1997) 153-172
21. Monma, C., Munson, B.S., Pulleyblank, W.R.: Minimum-weight two connected spanning networks. *Mathematical Programming* 46 (1990) 153-171
22. Rockafellar, R.T.: *Convex Analysis*. Princeton University Press (1970)
23. Steiglitz, K.S., Weinen, P., Kleitmann, D.S.: The design of minimum cost suivable networks. *IEEE Transactions and Circuit Theory* 16 (1969) 455-460
24. Stoer, M.: *Design of suivable Networks*. Lecture Notes in Mathematics Vol. 1531, Springer, Berlin (1992)
25. Winter, P.: Generalized Steiner Problem in Halin Graphs. *Proceedings of the 12th International Symposium on Mathematical Programming*, MIT (1985)
26. Winter, P.: Generalized Steiner Problem in series-parallel networks. *Journal of Algorithms* 7 (1986) 549-566

An Orientation Theorem with Parity Conditions

András Frank^{1*}, Tibor Jordán^{2**}, and Zoltán Szigeti³

¹ Department of Operations Research, Eötvös University,
H-1088 Rákóczi út 5., Budapest, Hungary, and
Ericsson Traffic Laboratory, H-1037 Laborc u. 1, Budapest, Hungary.
`frank@cs.elte.hu`

² BRICS[†], Department of Computer Science, University of Aarhus,
Ny Munkegade, building 540, DK-8000 Aarhus C, Denmark.
`jordan@daimi.au.dk`

³ Equipe Combinatoire, Université Paris VI,
4, place Jussieu, 75252 Paris, France.
`szigeti@ecp6.jussieu.fr`

Abstract. Given a graph $G = (V, E)$ and a set $T \subseteq V$, an orientation of G is called T -odd if precisely the vertices of T get odd in-degree. We give a good characterization for the existence of a T -odd orientation for which there exist k edge-disjoint spanning arborescences rooted at a prespecified set of k roots. Our result implies Nash-Williams' theorem on covering the edges of a graph by k forests and a (generalization of a) theorem due to Nebeský on upper embeddable graphs.

1 Introduction

Problems involving parity or connectivity conditions (for example matching theory or the theory of flows) form two major areas of combinatorial optimization. In some cases the two areas overlap. An example where parity and connectivity conditions are imposed simultaneously is the following theorem of Nebeský.

Theorem 1. [9] *A connected graph $G = (V, E)$ has a spanning tree F for which each connected component of $G - E(F)$ has an even number of edges if and only if*

$$|A| \geq c(G - A) + b(G - A) - 1 \quad (1)$$

holds for every $A \subseteq E$, where $c(G - A)$ denotes the number of connected components of $G - A$ and $b(G - A)$ denotes the number of those components D of $G - A$ for which $|V(D)| + |E(D)| - 1$ is odd. $\square$

* Supported by the Hungarian National Foundation for Scientific Research Grant OTKA T029772.

** Supported in part by the Danish Natural Science Research Council, grant no. 28808.

[†] Basic Research in Computer Science, Centre of the Danish National Research Foundation.

By a result of Jungerman [7] and Xuong [11], a graph G is *upper embeddable* if and only if it has a spanning tree F so that each connected component of $G - E(F)$ has an even number of edges. Thus Nebeský's theorem gives a good characterization for upper embeddable graphs. For more details on connections to topology see [6] or [9].

Theorem 1 can be reformulated in terms of orientations of G . Let $G = (V, E)$ be a connected undirected graph and let $T \subseteq V$. An orientation of G is called *T -odd* if precisely the vertices of T get odd in-degree. It is easy to see that G has a T -odd orientation if and only if $|E| + |T|$ is even. From this observation we obtain the following:

Proposition 1. *Let $G = (V, E)$ be a connected graph for which $|V| + |E| - 1$ is even. Then G has a spanning tree F for which each connected component of $G - E(F)$ has an even number of edges if and only if for every $r \in V$ there exists a $(V - r)$ -odd orientation of G which contains a spanning arborescence rooted at r . $\square$*

Motivated by Theorem 1 and Proposition 1, our goal in this paper is to investigate more general problems concerning orientations of undirected graphs simultaneously satisfying connectivity and parity requirements. Namely, given an undirected graph $G = (V, E)$, $T \subseteq V$ and $k \geq 0$, our main result gives a necessary and sufficient condition for the existence of a T -odd orientation of G which contains k edge-disjoint spanning arborescences rooted at a given set of k roots. This good characterization generalizes Theorem 1 and at the same time slightly simplifies condition (1). Furthermore, it implies Nash-Williams' theorem on covering the edges of a graph by k forests as well. These corollaries are discussed in Section 3. Note that a related problem on k -edge-connected T -odd orientations is solved in [5].

Our proof employs the proof method which was developed independently by Gallai and Anderson and which was first used to show an elegant proof for Tutte's theorem on perfect matchings of graphs, see [1]. In our case the weaker result the proof hinges on (which is Hall's theorem in the previously mentioned proof for Tutte's result) is an orientation theorem of the first author (Theorem 4 below).

Let $R = \{r_1, \dots, r_k\}$ be a *multiset* of vertices of G (that is, the elements of R are not necessarily pairwise distinct). By $T \oplus R$ we mean $((T \oplus r_1) \oplus \dots) \oplus r_k$, where $\oplus$ denotes the symmetric difference. For some $X \subseteq V$ the subgraph induced by X is denoted by $G[X]$. The number of edges in $G[X]$ is denoted by $i(X)$. For a partition $\mathcal{P} = \{V_1, \dots, V_t\}$ of V with t elements the set of edges connecting different elements of $\mathcal{P}$ is denoted by $E(\mathcal{P})$. We set $e(\mathcal{P}) = |E(\mathcal{P})|$. A subset F of edges of a directed graph D is a *spanning arborescence rooted at r* if F forms a spanning tree and each vertex has in-degree one except r . The *in-degree* of a set $X \subseteq V$ in a directed graph $D = (V, E)$ is denoted by $\rho(X)$. The following well-known result is due to Edmonds.

Theorem 2. [3] *Let R be a multiset of vertices of size k in a directed graph $D = (V, E)$. Then D contains k edge-disjoint spanning arborescences rooted at R if and only if $\rho(X) \geq k - |X \cap R|$ for every $X \subseteq V$. $\square$*

The following result is due to Frank.

Theorem 3. [4, Theorem 2.1] *Let $H = (V, E)$ be a graph, and let $g : V \rightarrow \mathbb{Z}_+$ be a function. Then there exists an orientation of H whose in-degree function ρ satisfies $\rho(v) = g(v)$ for every $v \in V$ if and only if the following two conditions hold.*

$$g(V) = |E| \quad (2)$$

$$g(X) \geq i(X) \quad \text{for every } \emptyset \neq X \subseteq V. \quad (3)$$

$\square$

We shall rely on the following orientation theorem, which is easy to prove from Theorem 3 and Theorem 2.

Theorem 4. *Let $H = (V, E)$ be a graph, let $R' = \{r'_1, \dots, r'_k\}$ be a multiset of k vertices of H and let $g : V \rightarrow \mathbb{Z}_+$ be a function. Then there exists an orientation of H whose in-degree function ρ satisfies $\rho(v) = g(v)$ for every $v \in V$ and for which there exist k edge-disjoint spanning arborescences with roots $\{r'_1, \dots, r'_k\}$ if and only if (2) and the following condition hold.*

$$g(X) \geq i(X) + k - |X \cap R'| \quad \text{for every } \emptyset \neq X \subseteq V. \quad (4)$$

$\square$

Note that R is a multiset in Theorem 4, hence by $|X \cap R|$ in (4) we mean $|\{r_i \in R : r_i \in X, i = 1, \dots, k\}|$. This convention will be used later on, whenever we take the intersection (or union) with a multiset.

Given $G = (V, E)$, $T \subseteq V$, $k \in \mathbb{Z}_+$ and a partition $\mathcal{P} = \{V_1, \dots, V_t\}$ of V , an element V_j ($1 \leq j \leq t$) is called *odd* if $|V_j \cap T| - i(V_j) - k$ is odd, otherwise V_j is *even*. The number of odd elements of $\mathcal{P}$ is denoted by $s_G(\mathcal{P}, T, k)$ (where some parameters may be omitted if they are clear from the context). Our main result is the following.

Theorem 5. *Let $G = (V, E)$ be a graph, $T \subseteq V$ and let $k \geq 0$ be an integer. For a multiset of k vertices $R = \{r_1, \dots, r_k\}$ of V there exists a $T \oplus R$ -odd orientation of G for which there exist k edge-disjoint spanning arborescences with roots $\{r_1, \dots, r_k\}$ if and only if*

$$e(\mathcal{P}) \geq k(t-1) + s(\mathcal{P}, T) \quad (5)$$

holds for every partition $\mathcal{P} = \{V_1, \dots, V_t\}$ of V .

2 The Proof of Theorem 5

Proof. (of Theorem 5) To see the necessity of condition (5), consider an orientation of G with the required properties and some partition $\mathcal{P} = \{V_1, \dots, V_t\}$ of V . The following fact is easy to observe.

Proposition 2. *For every $T \oplus R$ -odd orientation of G and for every V_j ($1 \leq j \leq t$) we have $|V_j \cap T| - i(V_j) - k \equiv \rho(V_j) - (k - |V_j \cap R|) \pmod{2}$.*

Proof. Since the orientation is $T \oplus R$ -odd, we obtain $\rho(V_j) + i(V_j) = \sum_{v \in V_j} \rho(v) \equiv |V_j \cap (T \oplus R)| \equiv |V_j \cap T| - |V_j \cap R|$ and the lemma follows. $\square$

Since there exist k edge-disjoint arborescences rooted at vertices of R , $\rho(V_j) - (k - |V_j \cap R|) \geq 0$ for each V_j . If this number is odd (or, equivalently by Proposition 2, if V_j is odd) then it is at least one. This yields $e(\mathcal{P}) = \sum_{V_j \in \mathcal{P}} \rho(V_j) = \sum_{V_j \in \mathcal{P}} (\rho(V_j) - (k - |V_j \cap R|)) + kt - |V \cap R| \geq s(\mathcal{P}, T) + kt - k = k(t-1) + s(\mathcal{P}, T)$, hence the necessity follows.

In what follows we prove that (5) is sufficient. An orientation is called *good* if the directed graph obtained contains k edge disjoint spanning arborescences rooted at R . Let us suppose that the statement of the theorem does not hold and let us take a counter-example (that is, a graph $G = (V, E)$ with T , R and k , for which (5) holds but no good $T \oplus R$ -odd orientation exists) for which $|V| + |E|$ is as small as possible.

Proposition 3. $e(\mathcal{P}) \equiv k(t-1) + s(\mathcal{P}, T) \pmod{2}$ for every partition $\mathcal{P}$.

Proof. By choosing $\mathcal{P}_0 = \{V\}$ in (5) we obtain that $|T| - |E| - k$ is even. This implies $s(\mathcal{P}, T) \equiv \sum_{j=1}^t (|V_j \cap T| - i(V_j) - k) = |T| - (|E| - e(\mathcal{P})) - kt \equiv |T| - |E| - k + e(\mathcal{P}) - k(t-1) \equiv e(\mathcal{P}) - k(t-1)$. $\square$

We call a partition $\mathcal{P}$ of V consisting of t elements *tight* if $e(\mathcal{P}) = k(t-1) + s(\mathcal{P}, T)$ and $t \geq 2$.

Lemma 1. *There exists a tight partition of V .*

Proof. Let ab be an arbitrary edge of G . Focus on the graph $G' = G - ab$ and the modified set $T' = T \oplus b$. If there was a good $T' \oplus R$ -odd orientation of G' then adding the arc ab would provide a good $T \oplus R$ -odd orientation of G , which is impossible. Thus, by the minimality of G , there exists a partition $\mathcal{P}$ of V consisting of t elements violating (5) in G' , that is, using Proposition 3, $e_{G'}(\mathcal{P}) + 2 \leq k(t-1) + s_{G'}(\mathcal{P}, T')$. Clearly, $t \geq 2$ holds.

For the same partition in G we have $e_G(\mathcal{P}) \leq e_{G'}(\mathcal{P}) + 1$ and also $s_G(\mathcal{P}, T) \geq s_{G'}(\mathcal{P}, T') - 1$, since adding the edge ab and replacing $T \oplus b$ by T may change the parity of at most one element of the partition. Thus $k(t-1) + s_G(\mathcal{P}, T) \geq k(t-1) + s_{G'}(\mathcal{P}, T') - 1 \geq e_{G'}(\mathcal{P}) + 2 - 1 \geq e_G(\mathcal{P}) \geq k(t-1) + s_G(\mathcal{P}, T)$, hence $\mathcal{P}$ is tight in G and the lemma follows. $\square$

Let us fix a tight partition $\mathcal{P} = \{V_1, \dots, V_t\}$ in G for which t is maximal. Denote the number of odd components of $\mathcal{P}$ by s .

Lemma 2. *Every element V_j of $\mathcal{P}$ ($1 \leq j \leq t$) has the following property:*

- (a) *if V_j is even, then (5) holds in $G[V_j]$, with respect to $T \cap V_j$,*
- (b) *if V_j is odd, then for each vertex $v \in V_j$, (5) holds in $G[V_j]$, with respect to $(T \cap V_j) \oplus v$.*

Proof. We handle the two cases simultaneously. Suppose that there exists a partition $\mathcal{P}'$ of t' elements in $G[V_j]$ violating (5) (with respect to $T \cap V_j$ in case (a) or with respect to $(T \cap V_j) \oplus v$, for some $v \in V_j$, in case (b)). By Proposition 3 this implies $k(t'-1) + s' \geq e(\mathcal{P}') + 2$, where s' denotes the number of odd elements of $\mathcal{P}'$. Consider the partition $\mathcal{P}'' = (\mathcal{P} - V_j) \cup \mathcal{P}'$, consisting of t'' elements from which s'' are odd. Clearly, $e(\mathcal{P}'') = e(\mathcal{P}) + e(\mathcal{P}')$ and $t'' = t + t' - 1$. Furthermore, $s'' \geq s + s' - 2$, since the parity of at most two elements may be changed (these are V_j – only in case (b) – and the element in $\mathcal{P}'$ which contains the vertex v – only in case (b) again). Since (5) holds for $\mathcal{P}''$ by the assumption of the theorem, we have $k(t''-1) + s'' \leq e(\mathcal{P}'') = e(\mathcal{P}) + e(\mathcal{P}') \leq k(t-1) + s + k(t'-1) + s' - 2 = k((t+t'-1)-1) + s + s' - 2 \leq k(t''-1) + s''$. Thus $\mathcal{P}''$ is a tight partition with $t'' > t$, which contradicts the choice of $\mathcal{P}$. $\square$

Let us denote the graph obtained from G by contracting each element V_j of $\mathcal{P}$ into a single vertex v_j ($1 \leq j \leq t$) by H . Let $R' = \{r'_1, \dots, r'_k\}$ denote the multiset of vertices of H corresponding to the vertices of R in G (that is, every root in some V_j yields a new root v_j). Furthermore, let A denote those vertices of H which correspond to odd elements of $\mathcal{P}$ and let $B = V(H) - A$. Note that since $\mathcal{P}$ is tight, we have

$$|E(H)| = e(\mathcal{P}) = k(t-1) + s(\mathcal{P}). \quad (6)$$

Now define the following function g on the vertex set of H .

$$g(v_j) = \begin{cases} k+1 - |V_j \cap R| & \text{if } v_j \in A \\ k - |V_j \cap R| & \text{otherwise.} \end{cases}$$

Lemma 3. *There exists an orientation of H whose in-degree function is g and which contains k edge-disjoint spanning arborescences with roots $\{r'_1, \dots, r'_k\}$.*

Proof. To prove the lemma we have to verify that the two conditions (2) and (4) of Theorem 4 are satisfied. First we can see that $g(V(H)) = g(A) + g(B) = s(k+1) + (t-s)k - k = k(t-1) + s = |E(H)|$ by the definition of g and by (6). Thus (2) is satisfied.

To verify (4), let us choose an arbitrary non-empty subset X of $V(H)$. Let us define the partition $\mathcal{P}^*$ of $V(G)$ by $\mathcal{P}^* := \{V_j : v_j \in V(H) - X\} \cup \bigcup_{v_j \in X} V_j$. Then $\mathcal{P}^*$ has $t^* = t - |X| + 1$ elements and the number of its odd elements s^* is at least $s - |X \cap A|$. Applying (5) for $\mathcal{P}^*$, it follows that $k(t^*-1) + s^* \leq e(\mathcal{P}^*)$.

Hence $k((t - |X| + 1) - 1) + s - |X \cap A| \leq k(t^* - 1) + s^* \leq e(\mathcal{P}^*) = e(\mathcal{P}) - i(X) = k(t - 1) + s - i(X)$. From this it follows that $i(X) + k \leq k|X| + |X \cap A|$. Therefore $i(X) + k - |X \cap R'| \leq k|X| + |X \cap A| - |X \cap R'| = |X \cap A|(k + 1) + |X \cap B|k - |X \cap R'| = g(X \cap A) + g(X \cap B) = g(X)$, proving that (4) is also satisfied. $\square$

Let us fix an orientation of H whose in-degree function $\rho_H = g$ and which contains a set $\mathcal{F}$ of k edge-disjoint spanning arborescences $\{F_1, \dots, F_k\}$ with roots $\{r'_1, \dots, r'_k\}$. Such an orientation exists by Lemma 3. Observe, that this orientation of H corresponds to a partial orientation of G (namely, an orientation of the edges of $E(\mathcal{P})$).

For any vertex v_j of H there are $g(v_j)$ arcs entering v_j . If V_j is even then each arc entering v_j belongs to some arborescence in $\mathcal{F}$. If V_j is odd then each arc entering v_j except exactly one belongs to some arborescence of $\mathcal{F}$, by the definition of g .

For an arbitrary $V_j \in \mathcal{P}$ let us denote by R_j^H the multiset of those vertices in V_j which are the heads of the arcs of this partial orientation entering V_j and belonging to some arborescence in $\mathcal{F}$. By the definition of g , we have $|R_j^H| = k - |V_j \cap R|$. Let $R_j = (V_j \cap R) \cup R_j^H$. Note that $|R_j| = |V_j \cap R| + |R_j^H| = k$. Furthermore, if V_j is odd then let us denote by a_j the vertex in V_j which is the head of the unique arc entering V_j and not belonging to any arborescence in $\mathcal{F}$. Let $T_j = T \cap V_j$ if V_j is even and let $T_j = (T \cap V_j) \oplus a_j$ if V_j is odd.

By the minimality of G and since $|V_j| < |V(G)|$ for each $1 \leq j \leq t$, Lemma 2 implies that for each j there exists a $T_j \oplus R_j$ -odd orientation of $G[V_j]$ which contains k edge-disjoint spanning arborescences with roots in R_j . Combining these orientations of the subgraphs induced by the elements of $\mathcal{P}$ and the orientation of $E(\mathcal{P})$ obtained earlier, we get an orientation of G . This orientation is clearly a good $T \oplus R$ -odd orientation of G , contradicting our assumption on G . This contradiction proves the theorem. $\square$

3 Corollaries

As we reformulated Theorem 1 in terms of odd orientations and spanning arborescences, we can similarly reformulate Theorem 5 in terms of even components and spanning trees.

Theorem 6. *A graph $G = (V, E)$ has k edge-disjoint spanning trees $F_1, \dots, F_k$ so that each connected component of $G - \cup_1^k E(F_i)$ has an even number of edges if and only if*

$$e(\mathcal{P}) \geq k(t - 1) + s \quad (7)$$

holds for each partition $\mathcal{P} = \{V_1, \dots, V_t\}$ of V , where s is the number of those elements of $\mathcal{P}$ for which $i(V_j) + k(|V_j| - 1)$ is odd.

Proof. As we observed, G has an orientation for which the in-degree of every vertex is even if and only if each connected component of G contains an even number of edges. Thus the desired spanning trees exist in G if and only if G has a $T \oplus R$ -odd

orientation which contains k edge-disjoint r -arborescences, where $T = V$, if k is odd, $T = \emptyset$, if k is even, and $R = \{r_1, \dots, r_k\}$, $r_i = r$ ($i = 1, \dots, k$) for an arbitrary $r \in V$. Based on this fact, Theorem 5 proves the theorem (by observing that (5) specializes to (7) due to the special choice of T). $\square$

The special case $k = 1$ of Theorem 6 corresponds to Theorem 1. Since (1) implies (7) if $k = 1$, Theorem 6 applies and we obtain a slightly simplified version of Nebeský's result. Note also that our main result provides a proof of different nature for Theorem 1 by using Theorem 4. Actually, Nebeský proved the defect form of the previous result, showing a min-max equality for the minimum number of components with odd number of edges of $G - E(F)$ among all possible spanning trees F of G (hence characterizing the *maximum genus* of the graph).

The next corollary we prove is Nash-Williams' classical theorem on forest covers.

Corollary 1. [8] *The edges of a graph $G = (V, E)$ can be covered by k forests if and only if*

$$i(X) \leq k(|X| - 1) \quad (8)$$

holds for every $\emptyset \neq X \subseteq V$.

Proof. We consider the sufficiency of the condition. Let $G = (V, E)$ be a graph for which (8) holds. The first claim is that we can add new edges to G until the number of edges equals $k(|V| - 1)$ without destroying (8). To see this, observe that the addition of a new edge $e = xy$ (which may be parallel to some other edges already present in G) cannot be added if and only if $x, y \in Z$ for some $Z \subseteq V$ with $i(Z) = k(|Z| - 1)$. Such a set, satisfying (8) with equality, will be called *full*. It is well-known that the function $i : 2^V \rightarrow Z_+$ is supermodular. Therefore for two intersecting full sets Z and W we have $k(|Z| - 1) + k(|W| - 1) = i(Z) + i(W) \leq i(Z \cap W) + i(Z \cup W) \leq k(|Z \cap W| - 1) + k(|Z \cup W| - 1) = k(|Z| - 1) + k(|W| - 1)$. Thus equality holds everywhere, and the sets $Z \cap W$ and $Z \cup W$ are also full. Now let F be a maximal full set (we may assume $F \neq V$) and $e = xy$ for some pair $x \in F, y \in V - F$. If we destroyed (8) by adding e , we would have a full set $x, y \in F'$ in G intersecting F , hence $F \cup F'$ would also be full by our previous observation. This contradicts the maximality of F .

Thus in the rest of the proof we may assume that $|E| = k(|V| - 1)$. We claim that there exist k edge-disjoint spanning trees in G . The existence of these trees immediately implies that G can be covered by k forests because $|E| = k(|V| - 1)$.

By Theorem 6, it is enough to prove that (7) holds in G . Let $\mathcal{P} = \{V_1, \dots, V_t\}$ be a partition of V and let $V_1, \dots, V_s$ denote the odd elements of $\mathcal{P}$ (with respect to k). Observe that for an odd element V_j the parity of $i(V_j)$ and $k(|V_j| - 1)$ must be different (this holds for even k and for odd k as well), hence these numbers cannot be equal. Thus we can count as follows: $e(\mathcal{P}) = |E| - \sum i(V_i) = k(|V| - 1) - \sum (i(V_i) : V_i \text{ is even}) - \sum (i(V_j) : V_j \text{ is odd}) \geq k(|V| - 1) - \sum (k(|V_i| - 1) : V_i \text{ is even}) - \sum (k(|V_j| - 1) - 1 : V_j \text{ is odd}) = k(|V| - 1) - k|V| + kt + s = k(t - 1) + s$, as required. $\square$

4 Remarks

The existence of a spanning tree of Theorem 1 with the required properties can be formulated as a *linear matroid parity* problem, hence it can be obtained from Lovász's characterization concerning the existence of a perfect matroid-matching [10] as well. (Although it is not an easy task to deduce Nebeský's result.) The reduction to a certain *co-graphic* matroid-parity problem was shown by Furst et al. [6]. Based on this, they developed a polynomial algorithm for the corresponding optimization problem. A similar reduction, where the matroid is the dual of the sum of k graphic matroids, works in the more general case of Theorem 6, too. However, from algorithmic point of view, such a reduction is not satisfactory, since it is not known how to represent the matroid in question.

Finally we remark that Proposition 1 was also observed independently by Chevalier et al. [2].

References

1. I. Anderson, Perfect matchings of a graph, J. Combin. Theory Ser. B, 10 (1971), 183-186.
2. O. Chevalier, F. Jaeger, C. Payan and N.H. Xuong, Odd rooted orientations and upper-embeddable graphs, Annals of Discrete Mathematics 17, (1983) 177-181.
3. J. Edmonds, Edge-disjoint branchings, in: R. Rustin (Ed.), Combinatorial Algorithms, Academic Press, (1973) 91-96.
4. A. Frank, Orientations of graphs and submodular flows, Congr. Numer. 113 (1996), 111-142.
5. A. Frank, Z. Király, Parity constrained k -edge-connected orientations, Proc. Seventh Conference on Integer Programming and Combinatorial Optimization (IPCO), Graz, 1999. LNCS, Springer, this issue.
6. M.L. Furst, J.L. Gross, and L.A. McGeoch, Finding a maximum genus graph imbedding, J. of the ACM, Vol. 35, No. 3, July 1988, 523-534.
7. M. Jungerman, A characterization of upper embeddable graphs, Trans. Amer. Math. Soc. 241 (1978), 401-406.
8. C. St. J. A. Nash-Williams, Edge-disjoint spanning trees of finite graphs, J. London Math. Soc. 36 (1961), 445-450.
9. L. Nebeský, A new characterization of the maximum genus of a graph, Czechoslovak Mathematical Journal, 31 (106) 1981, 604-613.
10. L. Lovász, Selecting independent lines from a family of lines in a space, Acta Sci. Univ. Szeged 42, 1980, 121-131.
11. N.H. Xuong, How to determine the maximum genus of a graph, J. Combin. Theory Ser. B 26 (1979), 217-225.

Parity Constrained k -Edge-Connected Orientations*

András Frank¹ and Zoltán Király²

¹ Department of Operations Research, Eötvös University, Rákóczi út 5, Budapest, Hungary, H-1088 and Ericsson Traffic Laboratory, Laborc u. 1, Budapest, Hungary, H-1037. frank@cs.elte.hu

² Department of Computer Science, Eötvös University, Rákóczi út 5, Budapest, Hungary, H-1088. kiraly@cs.elte.hu

Abstract. Parity (matching theory) and connectivity (network flows) are two main branches of combinatorial optimization. In an attempt to understand better their interrelation, we study a problem where both parity and connectivity requirements are imposed. The main result is a characterization of undirected graphs $G = (V, E)$ having a k -edge-connected T -odd orientation for every subset $T \subseteq V$ with $|E| + |T|$ even. (T -odd orientation: the in-degree of v is odd precisely if v is in T .) As a corollary, we obtain that every $(2k + 2)$ -edge-connected graph with $|V| + |E|$ even has a k -edge-connected orientation in which the in-degree of every node is odd. Along the way, a structural characterization will be given for digraphs with a root-node s having $k + 1$ edge-disjoint paths from s to every node and k edge-disjoint paths from every node to s .

1 Introduction

The notion of parity plays an important role in describing combinatorial structures. The prime example is W. Tutte's theorem [T47] on the existence of a perfect matching of a graph. Later, the notion of "odd components" has been extended and used by W. Mader [M78] in his disjoint A -paths theorem, by R. Giles [G82] in describing matching-forests, by L. Nebeský [N81] in determining the maximum genus, by W. Cunningham and J. Geelen [CG97] in characterizing optimal path-matchings. L. Lovász' [L80] general framework on matroid parity (as its name already suggests) also relies on odd components. Sometimes parity comes in already with the problem formulation. Lovász [L70] for example considered the existence of subgraphs with parity prescription on the degree of nodes. The theory of T -joins describes several problems of this type.

Another large class of combinatorial optimization problems concerns connectivity properties of graphs, in particular, the role of cuts, partitions, trees, paths, and flows are especially well studied.

* Research supported by the Hungarian National Foundation for Scientific Research Grant, OTKA T17580 and OTKA F014919. Part of the research was conducted while the first-named author was visiting EPFL (Lausanne, Switzerland, 1998).

In some cases the two areas overlap (for example, when we are interested in finding paths or subgraphs of given properties and the parity comes in the characterization). For example, Seymour's theorem [S81] on minimum T -joins implies a result on the edge-disjoint paths problem in planar graphs. In [FST84] some informal analogy was pointed out between results on parity and on connectivity, but in order to understand better the relationship of these two big aspects of combinatorial optimization, it is desirable to explore further problems where both parity and connectivity requirements are imposed. For example, Nebeský provided a characterization of graphs having an orientation in which every node is reachable from a given node by a directed path and the in-degree of every node is odd. Recently, [FJS98] extended this result to the case where, beside the parity constraints, the existence of k edge-disjoint paths was required from the specific node to every other one.

The goal of the present paper is to provide a new contribution to this broader picture. The main result is about orientations of undirected graphs simultaneously satisfying connectivity and parity requirements. The following concepts of connectivity will be used.

Let l be a positive integer. A digraph $D = (V, A)$ is **l -edge-connected** (**l -ec**, for short) if the **in-degree** $\varrho(X) = \varrho_D(X)$ of X , that is, the number of edges entering X is at least l for every non-empty proper subset X of V . The **out-degree** $\delta(X) = \delta_D(X)$ is the number of edges leaving X , that is, $\delta(X) = \varrho(V - X)$. The 1-ec digraphs are called **strongly connected**. A digraph D is said to be **l^- -edge-connected** (**l^- -ec**) if it has a node s , called **root node**, so that $\varrho(X) \geq l$ for every subset X with $\emptyset \subset X \subseteq V - s$, and $\varrho(X) \geq l - 1$ for every subset X with $s \in X \subset V$. When the role of the root is emphasized, we say that D is l^- -ec with respect to s . Throughout the root-node will be denoted by s . Note that by reorienting the edges of a directed path from s to another node t of an l^- -ec digraph one obtains an l^- -ec digraph with respect to root t .

Define a set-function p_l as follows. Let $p_l(\emptyset) := p_l(V) := 0$ and

$$p_l(X) := \begin{cases} l & \text{if } \emptyset \subset X \subseteq V - s \\ l - 1 & \text{if } s \in X \subset V. \end{cases} \quad (1.1)$$

By the definition, D is l^- -ec if and only if $\varrho(X) \geq p_l(X)$ holds for every $X \subseteq V$. By Menger's theorem the l^- -edge-connectivity of D is equivalent to requiring that D has l edge-disjoint paths from s to every node and $l - 1$ edge-disjoint paths from every node to s .

An undirected graph $G = (V, E)$ is **l -edge-connected** (**l -ec**) if the number $d(X)$ of edges connecting any non-empty proper subset X of V and its complement $V - X$ is at least l . G is called **$2l^-$ -edge-connected** (**$2l^-$ -ec**) if

$$e_G(\mathcal{F}) \geq lt - 1 \quad (1.2a)$$

$$\text{for every partition } \mathcal{F} := \{V_1, V_2, \dots, V_t\} \text{ of } V \quad (1.2b)$$

where $e_G(\mathcal{F})$ denotes the number of edges connecting distinct parts of $\mathcal{F}$. Throughout we will assume on partitions to admit at least two non-empty classes and no empty ones.

Note that a $2l$ -ec graph is automatically $2l^-$ -ec since $e_G(\mathcal{F}) = (\sum_i d(V_i))/2 \geq (2l)t/2 = lt$, that is, (1.2) is satisfied. Also, a $2l^-$ -ec graph G is $(2l-1)$ -ec since (1.2), when specialized to $t = 2$, requires for every cut of G to have at least $2l-1$ edges. In other words, $2l^-$ -edge-connectivity is somewhere between $(2l-1)$ -ec and $2l$ -ec.

Let T be a subset of V . We call T **G -even** if $|T| + |E|$ is even. An orientation of G is called **T -odd** if the in-degree of a node v is odd precisely if $v \in T$. It is easy to prove that a connected graph has a T -odd orientation if and only if T is G -even. (Namely, if an orientation is not yet T -odd, then there are at least two bad nodes. Let P be a path in the undirected sense that connects two bad nodes. By reversing the orientation of all of the edges of P , we obtain an orientation having two fewer bad nodes.) Note that if we subdivide each edge of G by a new node and let T' be denote the union of T and the set of subdividing nodes, then there is a one-to-one correspondence between T -odd orientations of G and T' -joins of the subdivided graph G' . (A **T' -join** is a subgraph of G' in which a node v is of odd degree precisely if v belongs to T' .)

As a main result, we will prove that an undirected graph G admits a k -ec T -odd orientation for every (!) G -even subset T of nodes if and only if G is $(2k+2)^-$ -ec. This is a co-NP characterization. An NP-characterization will also be derived by showing how these graphs can be built up from one node by repeated applications of two elementary graph-operations. It will also be shown that an undirected graph G has an l^- -ec orientation if and only if G is $2l^-$ -ec. Finally, a structural characterization of l^- -ec digraphs will be given and used for constructing all $2l^-$ -ec undirected graphs.

A remaining important open problem of the area, which has motivated the present investigations, is finding a characterization of graphs having a k -edge-connected orientation in which the in-degree of every node is of specified parity. No answer is known even in the special case $k = 1$.

The organization of the rest of the paper is as follows. The present section is completed by listing some definitions and notation. In Section 2 we formulate the new and the auxiliary results. Section 3 includes a characterization of l^- -ec digraphs. The last section contains the proof of the main theorem.

For two sets X and Y , $X - Y$ denotes the set of elements of X which are not in Y , and $X \oplus Y := (X - Y) \cup (Y - X)$ denotes the symmetric difference. A one-element set is called **singleton**. We often will not distinguish between a singleton and its element. In particular, the in-degree of a singleton $\{x\}$ will be denoted by $\varrho(x)$ rather than $\varrho(\{x\})$. For a set X and an element r , we denote $X \cup \{r\}$ by $X + r$.

For a directed or undirected graph G , let $i_G(X) = i(X)$ denote the number of edges having both end-nodes in X . Let $d(X, Y)$ (respectively, $\bar{d}(X, Y)$) denote

the number of edges connecting a node of $X - Y$ and a node of $Y - X$ (a node of $X \cap Y$ and a node of $V - (X \cup Y)$). Simple calculation yields the following identities for the in-degree function ϱ of a digraph G :

$$\varrho(X \cap Y) + \varrho(X \cup Y) = \varrho(X) + \varrho(Y) - d(X, Y), \quad (1.3a)$$

$$\varrho(X - Y) + \varrho(Y - X) = \varrho(X) + \varrho(Y) - \bar{d}(X, Y) - [\varrho(X \cap Y) - \delta(X \cap Y)]. \quad (1.3b)$$

Let f be an edge and r a node of G . Then $G - f$ and $G - r$ denote, respectively, the (di-)graphs arising from G by deleting edge f or node r .

Two subsets X and Y of node-set V are called **intersecting** if none of sets $X - Y, Y - X, X \cap Y$ is empty. If, in addition, $V - (X \cup Y)$ is non-empty, then X and Y are **crossing**. A family of subsets containing no two crossing (respectively, intersecting) sets is called **cross-free** (**laminar**).

Let p be a non-negative, integer-valued set-function on V for which $p(\emptyset) = p(V) = 0$. Function p is called **crossing supermodular** if $p(X) + p(Y) \leq p(X \cap Y) + p(X \cup Y)$ holds for every pair of crossing subsets X, Y of V . When this inequality is required only for crossing sets X, Y with $p(X) > 0$ and $p(Y) > 0$, we speak of **positively crossing supermodular** functions. A set-function p is called **monotone decreasing** if $p(X) \geq p(Y)$ holds whenever $\emptyset \neq X \subseteq Y$.

For a number x , let $x^+ := \max(0, x)$. For a function $m : V \rightarrow \mathbf{R}$ and subset $X \subseteq V$ we will use the notation $m(X) := \sum(m(v) : v \in X)$.

2 Results: New and Old

Our main result is as follows.

Theorem 2.1 *Let $G = (V, E)$ be an undirected graph with $n \geq 1$ nodes and let k be a positive integer. The following conditions are equivalent.*

- (1) *For every G -even subset $T \subseteq V$, G has a k -edge-connected T -odd orientation,*
- (2) *G is $(2k + 2)^-$ -edge-connected, that is,*

$$e_G(\mathcal{F}) \geq (k + 1)t - 1 \quad (2.1a)$$

$$\text{for every partition } \mathcal{F} := \{V_1, V_2, \dots, V_t\} \text{ of } V, \quad (2.1b)$$

- (3) *G can be constructed from a node by a sequence of the following two operations:*

- (i) *add an undirected edge connecting two (not necessarily distinct) existing nodes,*
- (ii) *choose a subset F of k (distinct) existing edges, subdivide each element of F by a new node, identify the k new nodes into one, denoted by r , and finally connect r with an arbitrary existing node u (that may or may not be an end-node of a subdivided edge).*

Since any l -ec graph is l^- -ec, we have the following corollary.

Corollary. *A $(2k+2)$ -edge-connected undirected graph $G = (V, E)$ with $|E| + |V|$ even has a k -edge-connected orientation so that the in-degree of every node is odd. $\square$*

We do not know any simple proof of this result even in the special case of $k = 1$. W. Mader [M82] proved a structural characterization of l -ec digraphs. We are going to show an analogous result for l^- -ec digraphs which will be used in the proof of Theorem 2.1 but may perhaps be interesting for its own sake, as well.

Theorem 2.2 *Let $D = (V, A)$ be a digraph and let $l \geq 2$ be an integer. Then D is l^- -edge-connected if and only if D can be constructed from a node by a sequence of the following two operations:*

(j) add a directed edge connecting two (not necessarily distinct) existing nodes,
 (jj) choose a set F of $l - 1$ (distinct) existing edges, subdivide each element of F by a new node, identify the $l - 1$ new nodes into one, denoted by r , and finally add a directed edge from an old node u (that may or may not be an end-node of a subdivided edge) to r .

C.St.J.A. Nash-Williams [N60] proved that an undirected graph G has an l -edge-connected orientation if and only if G is $2l$ -edge-connected. We will need the following auxiliary result.

Theorem 2.3 *An undirected graph G has an l^- -edge-connected orientation if and only if G is $2l^-$ -edge-connected.*

This result turns out to be an easy consequence of the following theorem.

Theorem 2.4 [F80] *Let $G = (V, E)$ be an undirected graph. Suppose that p is a non-negative integer-valued crossing supermodular set-function on V for which $p(\emptyset) = p(V) = 0$. Then there exists an orientation of G for which $\varrho(X) \geq p(X)$ holds for every $X \subseteq V$ if and only if both*

$$e_G(\mathcal{F}) \geq \sum_i p(V_i) \quad (2.2a)$$

and

$$e_G(\mathcal{F}) \geq \sum_i p(V - V_i) \quad (2.2b)$$

hold for every partition $\mathcal{F} = \{V_1, \dots, V_i\}$ of V . If, in addition, p is monotone decreasing, then already (2.2a) is sufficient. $\square$

In the proof of Theorem 2.2 we will need another auxiliary result.

Theorem 2.5 [F94] *Let U be a ground-set, p a non-negative, integer-valued positively crossing supermodular set-function on U for which $p(\emptyset) = p(U) = 0$.*

Let m_i, m_o be two non-negative integer-valued functions on U for which $m_i(U) = m_o(U)$. There exists a digraph $H = (U, F)$ for which

$$\varrho_H(X) \geq p(X) \text{ for every } X \subseteq U \quad (2.3)$$

and

$$\varrho_H(v) = m_i(v), \delta_H(v) = m_o(v) \text{ for every } v \in U \quad (2.4)$$

if and only if

$$m_i(X) \geq p(X) \text{ for every } X \subseteq U \quad (2.5a)$$

and

$$m_o(U - X) \geq p(X) \text{ for every } X \subseteq U. \quad (2.5b)$$

In Theorem 2.1 property (2) may be viewed as a co-NP characterization while (3) is an NP-characterization of property (1). The following result provides two other equivalent characterizations.

Theorem 2.6 *Let $G = (V, E)$ be an undirected graph with $n \geq 1$ nodes and let k be a positive integer. The following conditions are equivalent.*

- (1) *For every G -even subset T of V , G has a k -edge-connected T -odd orientation,*
- (4) *G has an orientation which is $(k+1)^-$ -edge-connected,*
- (5) *$G - J$ contains $k+1$ edge-disjoint spanning trees for every choice of a k -element subset J of edges.*

Proof. By Theorem 2.1, (1) is equivalent to (2) which in turn, by Theorem 2.3, is equivalent to (4). (5) clearly implies (2). To see that (2) implies (5), we apply Tutte's theorem on disjoint spanning trees [T61] which asserts that a graph $G = (V, E)$ contains k' edge-disjoint spanning trees if and only if $e_G(\mathcal{F}) \geq k't - k'$ holds for every partition $\mathcal{F} := \{V_1, V_2, \dots, V_t\}$ of V . Applying this result to $k' = k+1$, one can observe that (2) in Theorem 2.1 is equivalent to (5). $\square$

We remark that (5) can be derived directly from (3) without invoking Tutte's theorem. It is tempting to try to find a direct, short proof of the equivalence of (1) and (4), or at least one of the two opposite implications. We did not succeed even in the special case $k = 1$.

3 l^- -Edge-Connected Digraphs

Let $D = (V, A)$ be a digraph which is l^- -ec with respect to root-node s , that is, $\varrho(X) \geq p_l(X)$ for every $X \subset V$ where p_l is defined in (1.1). We call a set **tight** if $\varrho(X) = p_l(X)$. A node r of D and the set $\{r\}$ as well will be called **special** if $\varrho(r) = l = \delta(r) + 1$. (Since $\delta(s) \geq l$, s is not special.)

Lemma 3.1 *Suppose that a digraph $D = (V, A)$ with $|V| \geq 2$ is l^- -ec where $l \geq 2$. Then there is an edge $f = ur$ of D which does not enter any non-special tight set.*

Proof. We need some preparatory claims.

Claim A For crossing sets X, Y , one has $p_l(X) + p_l(Y) = p_l(X \cap Y) + p_l(X \cup Y)$ and $p_l(X) + p_l(Y) \leq p_l(X - Y) + p_l(Y - X)$. $\square$

Claim B The intersection of two crossing tight sets X, Y is not special.

Proof. By (1.3b) we have $\varrho(X) + \varrho(Y) = p_l(X) + p_l(Y) \leq p_l(X - Y) + p_l(Y - X) \leq \varrho(X - Y) + \varrho(Y - X) = \varrho(X) + \varrho(Y) - d(X, Y) - [\varrho(X \cap Y) - \delta(X \cap Y)]$ from which $\varrho(X \cap Y) = \delta(X \cap Y)$ follows and hence $X \cap Y$ cannot be special. $\square$

Claim C For two crossing tight sets X, Y , both $X \cap Y$ and $X \cup Y$ are tight. Moreover, $d(X, Y) = 0$ holds.

Proof. By (1.3a) we have $\varrho(X) + \varrho(Y) = p_l(X) + p_l(Y) = p_l(X \cap Y) + p_l(X \cup Y) \leq \varrho(X \cap Y) + \varrho(X \cup Y) = \varrho(X) + \varrho(Y) - d(X, Y)$ from which equality holds everywhere, and the claim follows. $\square$

Let us turn to the proof of the lemma and suppose indirectly that there is a family $\mathcal{T}$ of (distinct) non-special tight sets so that every edge of D enters a member of $\mathcal{T}$, and assume that $\mathcal{T}$ maximizes $\sum(|X|^2 : X \in \mathcal{T})$.

Claim D $\mathcal{T}$ contains no two crossing members.

Proof. Suppose indirectly that X and Y are two crossing members of $\mathcal{T}$. By Claim B, $X \cap Y$ is not special. By Claim C, $X \cap Y$ and $X \cup Y$ are tight. Hence $\mathcal{T}' := \mathcal{T} - \{X, Y\} \cup \{X \cap Y, X \cup Y\}$ is a family of non-special tight sets. Since $d(X, Y) = 0$, every edge of D enters a member of $\mathcal{T}'$, as well. However this contradicts the maximal choice of $\mathcal{T}$ since $|X|^2 + |Y|^2 < |X \cap Y|^2 + |X \cup Y|^2$. $\square$

Let $\mathcal{K} := \{X \in \mathcal{T} : s \notin X\}$ and $\mathcal{L} := \{V - X : X \in \mathcal{T}, s \in X\}$. Then $\mathcal{K}$ contains no special set, $\varrho(X) = l$ for every $X \in \mathcal{K}$, $\delta(X) = l - 1$ for every $X \in \mathcal{L}$. Let $\mathcal{C}$ denote the union of $\mathcal{K}$ and $\mathcal{L}$ in the sense that if X is a set belonging to both $\mathcal{K}$ and $\mathcal{L}$, then $\mathcal{C}$ includes two copies of X . Now $\mathcal{C}$ is a laminar family of subsets of $V - s$, and every edge e of D is **covered** by $\mathcal{C}$ in the sense that e enters a member of $\mathcal{K}$ or leaves a member of $\mathcal{L}$. Let us choose families $\mathcal{K}$ and $\mathcal{L}$ so as to satisfy all these properties and so that $\sum(|X| : X \in \mathcal{C})$ is minimum.

Claim E There is no node $v \in V - s$ for which $\{v\} \in \mathcal{K}$ and $\{v\} \in \mathcal{L}$.

Proof. $v \in \mathcal{L}$ implies $\delta(v) = l - 1$. $v \in \mathcal{K}$ implies $\varrho(v) = l$, that is, v would be special, contradicting the assumption on $\mathcal{K}$. $\square$

We distinguish two cases.

Case 1 Every member of $\mathcal{C}$ is a singleton.

Let $K = \{v : \{v\} \in \mathcal{K}\}$. Since D is strongly connected and $|V| \geq 2$, there is an edge $e = st$ leaving s . Edge e cannot leave any member of $\mathcal{L}$ since these members are subsets of $V - s$. Therefore e must enter a member of $\mathcal{K}$, that is, K is non-empty. By the strong connectivity of D , there is an edge e' leaving K . By Claim E, no element of K , as a singleton, is a member of $\mathcal{L}$, and hence edge e' neither enters a member of $\mathcal{K}$ nor leaves a member of $\mathcal{L}$, contradicting the assumption. Therefore Case 1 cannot occur.

Case 2 *There is a non-singleton member Z of $\mathcal{C}$.*

Suppose that Z is minimal with respect to containment.

Claim F *Z induces a strongly connected digraph.*

Proof. Suppose indirectly that there is a subset $\emptyset \subset X \subset Z$ so that there is no edge in D from X to $Z - X$. If $Z \in \mathcal{K}$, then replace Z in $\mathcal{K}$ by $Z - X$. Since $l \leq \varrho(Z - X) \leq \varrho(Z) = l$ we have $\varrho(Z - X) = l$. Furthermore, $Z - X$ cannot be special since every edge entering Z enters $Z - X$ as well and hence every edge entering X leaves $Z - X$ from which $l \leq \varrho(X) \leq \delta(Z - X)$.

If $Z \in \mathcal{L}$, then replace Z in $\mathcal{L}$ by X . In both cases we obtain a laminar family satisfying the requirements for $\mathcal{C}$ and this contradicts the minimal choice of $\mathcal{C}$. $\square$

Subcase 2.1 $Z \in \mathcal{K}$.

There must be an element v of Z for which $\{v\} \notin \mathcal{K}$, for otherwise Z can be left out from $\mathcal{K}$. We claim that $\{v\} \notin \mathcal{K}$ for every $v \in Z$. For otherwise $X := \{x \in Z : \{x\} \in \mathcal{K}\}$ is a non-empty, proper subset of Z , so by Claim F there is an edge $e = xy$ with $x \in X, y \in Z - X$, and then e cannot be covered by $\mathcal{C}$ (using that, by Claim E, $\{x\}$ is not in $\mathcal{L}$).

It follows that every edge uv with $u, v \in Z$ leaves a member of $\mathcal{L}$ which is a singleton, by the minimal choice of Z , and hence, by Claim F, $\{v\}$ is in $\mathcal{L}$ for every $v \in Z$. Then we have $l = \varrho(Z) = \sum(\varrho(v) : v \in Z) - i(Z) \geq l|Z| - i(Z)$ and $l - 1 \leq \delta(Z) = \sum(\delta(v) : v \in Z) - i(Z) = (l - 1)|Z| - i(Z)$ from which $(l - 1)(|Z| - 1) \geq i(Z) \geq l(|Z| - 1)$, a contradiction. Therefore Subcase 2.1 cannot occur.

Subcase 2.2 $Z \in \mathcal{L}$.

There must be an element v of Z for which $\{v\} \notin \mathcal{L}$, for otherwise Z can be left out from $\mathcal{L}$. We claim that $\{v\} \notin \mathcal{L}$ for every $v \in Z$. For otherwise $X := \{x \in Z : \{x\} \in \mathcal{L}\}$ is a non-empty, proper subset of Z , so by Claim F there is an edge $e = yx$ with $x \in X, y \in Z - X$, and then e cannot be covered by $\mathcal{C}$ (using that, by Claim E, $\{x\}$ is not in $\mathcal{K}$).

It follows that every edge uv with $u, v \in Z$ must enter a member of $\mathcal{K}$, which is a singleton, by the minimal choice of Z , and hence, by Claim F, $\{v\}$ is in $\mathcal{K}$ for every $v \in Z$. Therefore, as $\mathcal{K}$ contains no special members, no element of Z is special, from which $\delta(v) \geq l$ holds for every $v \in Z$.

We have $l - 1 = \delta(Z) = \sum(\delta(v) : v \in Z) - i(Z) \geq l|Z| - i(Z)$ and $l \leq \varrho(Z) = \sum(\varrho(v) : v \in Z) - i(Z) = l|Z| - i(Z)$ from which $l - 1 \geq l$, a contradiction. Therefore Subcase 2.2 cannot occur either, and this completes the proof of Lemma 3.1. $\square$

Proof of Theorem 2.2. It is easy to see that both operations (j) and (jj) given in the theorem preserve l^- -edge-connectivity.

To see the converse, suppose that D is l^- -ec. If D has no edges, then D has the only node s . Suppose now that A is non-empty and assume by induction

that every l^- -ec digraph, having a fewer number of edges than D has, is **constructible** in the sense that it can be constructed as described in the theorem.

If D has an edge f so that $D' = D - f$ is l^- -ec, then D' is constructible and then we obtain D from D' by adding back f , that is, by operation (j). Therefore we may assume that the deletion of any edge destroys l^- -edge-connectivity.

By Lemma 3.1 there is an edge $f = ur$ of D so that r is special and so that $\varrho'(X) \geq p_l(X)$ for every subset $X \subseteq V$ distinct from $\{r\}$ where ϱ' denotes the in-degree function of digraph $D' := D - f$. Since r is special, we have $\varrho'(r) = l - 1 = \delta'(r)$ where δ' is the out-degree function of D' .

Let $D_1 = (U, A_1)$ be the digraph arising from D by deleting r (where $U := V - r$), and let ϱ_1 denote the in-degree function of D_1 . Let $m_i(u)$ (respectively, $m_o(u)$) denote the number of parallel edges in D' from r to u (from u to r). From $\varrho'(r) = \delta'(r)$ we have $m_o(U) = m_i(U)$. Let $p(X) := (p_l(X) - \varrho_1(X))^+ \ (X \subset U)$ and $p(\emptyset) := p(U) := 0$. Since both p_l and $-\varrho_1$ are crossing supermodular, p is positively crossing supermodular.

We claim that (2.5a) holds. Indeed, for every $\emptyset \subset X \subset U$ one has $p_l(X) \leq \varrho'(X) = \varrho_1(X) + m_i(X)$ from which $p(X) = (p_l(X) - \varrho_1(X))^+ \leq m_i(X)$, which is (2.5a).

We claim that (2.5b) holds, as well. Indeed, for every $\emptyset \subset X \subset U$ we have $p_l(X) = p_l(X + r) \leq \varrho'(X + r) = \varrho_1(X) + m_o(U - X)$ from which $p(X) = (p_l(X) - \varrho_1(X))^+ \leq m_o(U - X)$, which is (2.5b).

By Theorem 2.5, there exists a digraph $H = (U, F)$ satisfying (2.3) and (2.4). It follows from (2.3) and from the definition of p that the digraph $D_1 + H := (U, A_1 \cup F)$ is l^- -ec. Then $D_1 + H$ is constructible by induction, as $|A_1 \cup F| = |A| - (2l - 1) + (l - 1) < |A|$. By (2.4), D arises from $D_1 + H$ by applying operation (jj) with the choice F , proving that D is also constructible. $\square$

Remark.

The proof of Theorem 2.5 in [F94] is algorithmic and gives rise to a combinatorial strongly polynomial algorithm if an oracle for handling p is available. We applied Theorem 2.5 to a function p defined by $p(X) := (p_l(X) - \varrho_1(X))^+$ and in this case the oracle can indeed be constructed via a network flow algorithm. Hence we can find in polynomial time a digraph $H = (U, F)$ for which $D_1 + H$ is l^- -ec. By applying this method at most $|A|$ times one can find the sequence of operations (j) and (jj) guaranteed by Theorem 2.2.

4 Proof of Theorem 2.1

(1) $\rightarrow$ (2). Let $\mathcal{F} := \{V_1, \dots, V_t\}$ be a partition of V . For $j = 2, \dots, t$ choose an element t_j of V_j if $k + i(V_j)$ is even. Furthermore, if the number of chosen elements plus $|E|$ is odd, then choose an element t_1 of V_1 . Let T be the set of chosen elements. Then T is G -even, and by (1), G has a k -ec T -odd orientation. For every $j = 2, \dots, t$, $\varrho(V_j) \geq k$, and we claim that equality cannot occur. Indeed, if $k + i(V_j) = \varrho(V_j) + i(V_j) = \sum(\varrho(v) : v \in V_j) \equiv |V_j \cap T| \pmod{2}$, then $k + i(V_j) + |V_j \cap T|$ would be even contradicting the definition of T .

Therefore, for every $j = 2, \dots, t$ we have $\varrho(V_j) \geq k+1$ and also $\varrho(V_1) \geq k$. Hence $e_G(\mathcal{F}) = \sum_{j=1}^t \varrho(V_j) \geq (k+1)(t-1) + k = (k+1)t - 1$, that is, (2.1a) holds.

(2) $\rightarrow$ (3) Let s be any node of G . By Theorem 2.3, G has a $(k+1)^-$ -ec orientation, denoted by $D = (V, A)$. By Theorem 2.2, D can be constructed from s by a sequence of operations (j) and (jj). The corresponding sequence of operations (i) and (ii) provides G .

(3) $\rightarrow$ (1). We use induction on the number of edges. There is nothing to prove if G has no edges so suppose that E is non-empty. Let T be a G -even subset of V . Let G' denote the graph from which G is obtained by one of the operations (i) and (ii). By induction, we may assume that G' has a k -ec T' -odd orientation for every G' -even set T' .

Suppose first that G arises from G' by adding a new edge $f = xy$. Let $T' := T \oplus \{y\}$. Clearly, T' is G' -even. By induction, there exists a k -ec T' -odd orientation of G' . By orienting edge e from x to y , we obtain a k -ec T -odd orientation of G .

Second, suppose that G arises from G' by operation (ii). In case $r \in T$, define $T' := T - r$ if k is even and $T' := (T - r) \oplus \{u\}$ if k is odd. In case $r \notin T$, define $T' := T \oplus \{u\}$ if k is even and $T' = T$ if k is odd. Then T' is G' -even and, by induction, G' has a k -ec T' -odd orientation. Orient the undirected edge ur from u to r if either k is even and $r \in T$ or else k is odd and $r \notin T$. Otherwise orient ur from r to u .

Recall that F denotes the subset of edges of G' used in Operation (jj). Orient the undirected edge ur from r to u if either k is odd and $r \in T$ or else k is even and $r \notin T$. Furthermore, if an element xy of F gets orientation from x to y , then let the two corresponding edges of G be oriented from x to r and from r to y , respectively. Obviously, the resulting orientation of G is k -ec and T -odd. $\square$

Remark.

Condition (2) in Theorem 2.1 shows that the property in (1) is in co-NP. Indeed, if (2) fails to hold for a partition, then a G -even subset T can be constructed (as was proved in (1) $\rightarrow$ (2)) for which no k -ec T -odd orientation exists.

For a given graph G the question whether there is a partition $\mathcal{F}$ violating (2.1a) or G can be constructed as described in condition (3) of Theorem 2.1 can be decided algorithmically as follows. The proof of Theorem 2.3 described in [F80] is algorithmic, and gives rise to a combinatorial strongly polynomial time algorithm in the special case $p = p_l$ for finding either a l^- -ec orientation of G or else a partition violating (1.2a) (which is equivalent to (2.1a) for $l = k+1$). As we have mentioned at the end of Section 3, finding a sequence of operations (j) and (jj) to build D , and hence a sequence of operations (i) and (ii) to build G , can also be done in polynomial time.

Naturally, this is just a rough proof of the existence of a combinatorial polynomial time algorithm for finding either a partition of V violating (2.1a) or else a sequence of operations (i) and (ii) to build G , and leaves room for improvements to get a decent bound on the complexity.

References

- [CG97] W. Cunningham and J. Geelen: *The optimal path-matching problem*, Combinatorica, Vol. 17, No. 3 (1997) pp. 315-338.
- [F80] A. Frank: *On the orientation of graphs*, J. Combinatorial Theory, Ser. B., Vol. 28, No. 3 (1980) pp. 251-261.
- [FST84] A. Frank, A. Sebő and É. Tardos: *Covering directed and odd cuts*, Mathematical Programming Studies 22 (1984) pp. 99-112.
- [FJS98] A. Frank, T. Jordán and Z. Szigeti: *An orientation theorem with parity constraints*, to appear in IPCO '99.
- [F94] A. Frank: *Connectivity augmentation problems in network design*, in: Mathematical Programming: State of the Art 1994, eds. J.R. Birge and K.G. Murty, The University of Michigan, pp. 34-63.
- [G82] R. Giles: *Optimum matching forests, I-II-III*, Mathematical Programming, 22 (1982) pp. 1-51.
- [L70] L. Lovász: *Subgraphs with prescribed valencies*, J. Combin. Theory, 8 (1970) pp. 391-416.
- [L80] L. Lovász: *Selecting independent lines from a family of lines in a space*, Acta Sci. Univ. Szeged, 43 (1980) pp. 121-131.
- [M78] W. Mader: *Über die Maximalzahl kantendisjunkter A-Wege*, Archiv der Mathematik (Basel) 30 (1978) pp. 325-336.
- [M82] W. Mader: *Konstruktion aller n -fach kantenzusammenhängenden Digraphen*, Europ. J. Combinatorics 3 (1982) pp. 63-67.
- [N60] C.St.J.A. Nash-Williams: *On orientations, connectivity, and odd vertex pairings in finite graphs*, Canad. J. Math. 12 (1960) pp. 555-567.
- [N81] L. Nebeský: *A new characterization of the maximum genus of a graph*, Czechoslovak Mathematical Journal, 31 (106), (1981) pp. 604-613.
- [S81] P.D. Seymour: *On odd cuts and plane multicommodity flows*, Proceedings of the London Math. Soc. 42 (1981) pp. 178-192.
- [T61] W.T. Tutte: *On the problem of decomposing a graph into n connected factors*, J. London Math. Soc. 36 (1961) pp. 221-230.
- [T47] W.T. Tutte: *The factorization of linear graphs*, J. London Math. Soc. 22 (1947) pp. 107-111.

Approximation Algorithms for MAX 4-SAT and Rounding Procedures for Semidefinite Programs

Eran Halperin and Uri Zwick

Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel.
{heran,zwick}@math.tau.ac.il

Abstract. Karloff and Zwick obtained recently an optimal $7/8$ -approximation algorithm for MAX 3-SAT. In an attempt to see whether similar methods can be used to obtain a $7/8$ -approximation algorithm for MAX SAT, we consider the most natural generalization of MAX 3-SAT, namely MAX 4-SAT. We present a semidefinite programming relaxation of MAX 4-SAT and a new family of rounding procedures that try to cope well with clauses of various sizes. We study the potential, and the limitations, of the relaxation and of the proposed family of rounding procedures using a combination of theoretical and experimental means. We select two rounding procedures from the proposed family of rounding procedures. Using the first rounding procedure we seem to obtain an *almost optimal* 0.8721 -approximation algorithm for MAX 4-SAT. Using the second rounding procedure we seem to obtain an *optimal* $7/8$ -approximation algorithm for *satisfiable* instances of MAX 4-SAT. On the other hand, we show that no rounding procedure from the family considered can yield an approximation algorithm for MAX 4-SAT whose performance guarantee on all instances of the problem is greater than 0.8724 .

Although most of this paper deals specifically with the MAX 4-SAT problem, we believe that the new family of rounding procedures introduced, and the methodology used in the design and in the analysis of the various rounding procedures considered would have a much wider range of applicability.

1 Introduction

MAX SAT is one of the most natural optimization problems. An instance of MAX SAT in the Boolean variables $x_1, \dots, x_n$ is composed of a collection of clauses. Each clause is the disjunction of an arbitrary number of literals. Each literal is a variable, x_i , or a negation, $\bar{x}_i$, of a variable. Each clause has a non-negative weight w associated with it. The goal is to find a 0-1 assignment of values to the Boolean variables $x_1, \dots, x_n$ so that the sum of the weights of the satisfied clauses is maximized.

Following a long line of research by many authors, we now know that MAX SAT is APX-hard (or MAX SNP-hard) [21,10,3,2,7,6,18]. This means that there is a constant $\epsilon > 0$ such that, assuming $P \neq NP$, there is no polynomial time approximation algorithm for MAX SAT with a performance guarantee of at

least $1 - \epsilon$ on all instances of the problem. Approximation algorithms for MAX SAT were designed by many authors, including [16,28,11,12,5,4]. The best performance ratio known for the problem is currently 0.77 [4]. An approximation algorithm for MAX SAT with a *conjectured* performance guarantee of 0.797 is given in [31].

In a major breakthrough, Håstad [14] showed recently that no polynomial time approximation algorithm for MAX SAT can have a performance guarantee of more than $7/8$, unless $P=NP$. Håstad's shows, in fact, that no polynomial time approximation algorithm for satisfiable instances of MAX $\{3\}$ -SAT can have a performance guarantee of more than $7/8$. MAX $\{3\}$ -SAT is the subproblem of MAX SAT in which each clause is of size *exactly* three. An instance is satisfiable if there is an assignment that satisfies all its clauses.

Karloff and Zwick [17] obtained recently an optimal $7/8$ -approximation algorithm for MAX 3-SAT, the version of MAX SAT in which each clause is of size *at most* three. (This claim appears in [17] as a conjecture. It has since been proved.) Their algorithm uses semidefinite programming. A much simpler approximation algorithm has a performance guarantee of $7/8$ if all clauses are of size *at least* three. If all clauses are of size at least three then a *random* assignment satisfies, on the average, at least $7/8$ of the clauses.

We thus have a performance guarantee of $7/8$ for instances in which all clauses are of size at most three, and for instances in which all clauses are of size at least three. Can we get a performance guarantee of $7/8$ for *all* instances of MAX SAT? In an attempt to answer this question, we check the prospects of obtaining a $7/8$ -approximation algorithm for MAX 4-SAT, the subproblem of MAX SAT in which each clause is of size *at most* four. As it turns out, this is already a challenging problem.

The $7/8$ -approximation algorithm for MAX 3-SAT starts by solving a semidefinite programming relaxation of the problem. It then *rounds* the solution of this program using a random hyperplane passing through the origin. It is natural to try to obtain a similar approximation algorithm for MAX 4-SAT. It is not difficult, see Section 2, to obtain a semidefinite programming relaxation of MAX 4-SAT. It is again natural to try to round this solution using a random hyperplane. It turns out, however, that the performance guarantee of this algorithm is only 0.845173. Although this is much better than all previous performance guarantees for MAX 4-SAT, this guarantee is, unfortunately, below $7/8$.

As the semidefinite programming relaxation of MAX 4-SAT is the strongest relaxation of its kind (see again Section 2), it seems that a different rounding procedure should be used. We describe, in Section 3, a new family of rounding procedures. This family extends all the families of rounding procedures previously suggested for maximum satisfiability problems. The difficulty in developing good rounding procedures for MAX 4-SAT is that rounding procedures that work well for the short clauses, do not work so well for the longer clauses, and vice versa. Rounding procedures from the new family try to work well on all clause sizes simultaneously. We initially hoped that an appropriate rounding procedure from this family could be used to obtain $7/8$ -approximation algorithms for MAX

4-SAT and perhaps even MAX SAT. It turns out, however, that the new family falls just short of this mission. The experiments that we have made suggest that a rounding procedure from the family, which we explicitly describe, can be used to obtain a 0.8721-approximation algorithm for MAX 4-SAT. Unfortunately, no rounding procedure from the family yields an approximation algorithm for MAX 4-SAT with a performance guarantee larger than 0.8724.

We have more success with MAX $\{2, 3, 4\}$ -SAT, the version of MAX SAT in which the clauses are of size two, three or four. We present a second rounding procedure from the family that seems to yield an optimal $7/8$ -approximation algorithm for MAX $\{2, 3, 4\}$ -SAT. A $7/8$ -approximation algorithm for MAX $\{2, 3, 4\}$ -SAT yields immediately an optimal $7/8$ -approximation algorithm for *satisfiable* instances of MAX 4-SAT, as clauses of size one can be easily eliminated from satisfiable instances.

To determine the performance guarantee obtained using a given rounding procedure R , or at least a lower bound on this ratio, we have to find the global minimum of a function $ratio_R(v_0, v_1, v_2, v_3, v_4)$, given a set of constraints on the unit vectors $v_0, v_1, \dots, v_4 \in \mathbb{R}^5$. The function $ratio_R$ is a fairly complicated function determined by the rounding procedure R . As five unit vectors are determined, up to rotations, by the $\binom{5}{2} = 10$ angles between them, the function $ratio_R$ is actually a function of 10 real variables. Finding the global minimum of $ratio_R$ analytically is a formidable task. In the course of our investigation we experimented with hundreds of rounding procedures. Finding these minima ‘by hand’ was not really an option. We have implemented a set of Matlab functions that use numerical techniques to find these minima.

The discussion so far centered on the quality of the rounding procedures considered. We also consider the quality of the suggested semidefinite programming relaxation itself. The integrality ratio of the MAX 4-SAT relaxation cannot be more than $7/8$, as it is also a relaxation of MAX 3-SAT. We also show that the integrality ratio of the relaxation, considered as a relaxation of the problem MAX $\{1, 4\}$ -SAT, is at most 0.8753. The fact that this ratio is, at best, just above $7/8$ is another indication of the difficulty of obtaining optimal $7/8$ -approximation algorithm for MAX 4-SAT and MAX SAT. It may also indicate that a stronger semidefinite programming relaxation would be needed to accomplish this goal.

The fact that numerical optimization techniques were used to compute the performance guarantees of the algorithms means that we cannot claim the existence of a 0.8721-approximation algorithm for MAX 4-SAT, and of a $7/8$ -approximation algorithm for MAX $\{2, 3, 4\}$ -SAT as theorems. We believe, however, that it is possible to prove these claims analytically and promote them to the status of theorems, as was eventually done with the optimal $7/8$ -approximation algorithm for MAX 3-SAT. This would require, however, considerable effort. It may make more sense, therefore, to look for an approximation algorithm that seems to be a $7/8$ -approximation algorithm for MAX 4-SAT before proceeding to this stage.

In addition to implementing a set of Matlab functions that try to find the performance guarantee of a given rounding procedure from the family considered,

we have also implemented a set of functions that *search* for good rounding procedures. The whole project required about 3000 lines of code. The two rounding procedures mentioned above, and several other interesting rounding procedures mentioned in Section 5, were found automatically using this system, with some manual help from the authors. The total running time used in the search for good rounding procedures is measured by months.

We end this section with a short survey of related results. The 7/8-approximation algorithm for MAX 3-SAT is based on the MAX CUT approximation algorithm of Goemans and Williamson [12]. A 0.931-approximation algorithm for MAX 2-SAT was obtained by Feige and Goemans [9]. Asano [4] obtained a 0.770-approximation algorithm for MAX SAT. Trevisan [25] obtained a 0.8-approximation algorithm for *satisfiable* MAX SAT instances. The last two results are also the best published results for MAX 4-SAT.

2 Semidefinite Programming Relaxation of MAX 4-SAT

Karloff and Zwick [17] describe a canonical way of obtaining semidefinite programming relaxations for any constraint satisfaction problem. We now describe the canonical relaxation of MAX 4-SAT obtained using this approach.

Assume that $x_1, \dots, x_n$ are the variables of the MAX 4-SAT instance. We let $x_0 = 0$ and $x_{n+i} = \bar{x}_i$, for $1 \leq i \leq n$. The semidefinite program corresponding to the instance has a variable unit vector v_i , corresponding to each literal x_i , and scalar variables z_i, z_{ij}, z_{ijk} or z_{ijkl} corresponding to the clauses $x_i, x_i \vee x_j, x_i \vee x_j \vee x_k$ and $x_i \vee x_j \vee x_k \vee x_l$ of the instance, where $1 \leq i, j, k \leq 2n$. Note that all clauses, including those that contain negated literals, can be expressed in this form. Clearly, we require $v_{n+i} = -v_i$, or $v_i \cdot v_{n+i} = -1$, for $1 \leq i \leq n$.

The objective of the semidefinite program is to maximize the function

$$\sum_i w_i z_i + \sum_{i,j} w_{ij} z_{ij} + \sum_{i,j,k} w_{ijk} z_{ijk} + \sum_{i,j,k,l} w_{ijkl} z_{ijkl} ,$$

where the w_i 's, w_{ij} 's, w_{ijk} 's and w_{ijkl} 's are the non-negative weights of the different clauses, subject to the following collection of constraints. For ease of notation, we write down the constraints that correspond to the clauses $x_1, x_1 \vee x_2, x_1 \vee x_2 \vee x_3$ and $x_1 \vee x_2 \vee x_3 \vee x_4$. The constraints corresponding to the other clauses are easily obtained by plugging in the corresponding indices. The constraints corresponding to x_1 and $x_1 \vee x_2$ are quite simple:

$$z_1 = \frac{1-v_0 \cdot v_1}{2} \quad , \quad z_{12} \leq \frac{3-v_0 \cdot v_1 - v_0 \cdot v_2 - v_1 \cdot v_2}{4} \quad , \quad z_{12} \leq 1 .$$

The constraints corresponding to $x_1 \vee x_2 \vee x_3$ are slightly more complicated:

$$\begin{aligned} z_{123} &\leq \frac{4-(v_0+v_1) \cdot (v_2+v_3)}{4} \quad , \quad z_{123} \leq \frac{4-(v_0+v_2) \cdot (v_1+v_3)}{4} \\ z_{123} &\leq \frac{4-(v_0+v_3) \cdot (v_1+v_2)}{4} \quad , \quad z_{123} \leq 1 \end{aligned}$$

It is not difficult to check that the first three constraints above are equivalent to the requirement that

$$z_{123} \leq \frac{4 - (v_{i_0} \cdot v_{i_1} + v_{i_1} \cdot v_{i_2} + v_{i_2} \cdot v_{i_3} + v_{i_3} \cdot v_{i_0})}{4},$$

for any permutation i_0, i_1, i_2, i_3 on $0, 1, 2, 3$. We will encounter similar constraints for the 4-clauses. The constraints corresponding to $x_1 \vee x_2 \vee x_3 \vee x_4$ are even more complicated. For any permutation i_0, i_1, i_2, i_3, i_4 on $0, 1, 2, 3, 4$ we require:

$$z_{1234} \leq \frac{5 - (v_{i_0} \cdot v_{i_1} + v_{i_1} \cdot v_{i_2} + v_{i_2} \cdot v_{i_3} + v_{i_3} \cdot v_{i_4} + v_{i_4} \cdot v_{i_0})}{4},$$

$$z_{1234} \leq \frac{5 - (v_{i_0} + v_{i_4}) \cdot (v_{i_1} + v_{i_2} + v_{i_3}) + v_{i_0} \cdot v_{i_4}}{4}, \quad z_{1234} \leq 1.$$

The first line above contributes 12 different constraints, the second line contributes 10 different constraints. Together with the constraint $z_{1234} \leq 1$ we get a total of 23 constraints per 4-clause. In addition, for every distinct $0 \leq i_1, i_2, i_3, i_4, i_5 \leq 2n$, we require

$$\sum_{1 \leq j < k \leq 3} v_{i_j} \cdot v_{i_k} \geq -1 \quad \text{and} \quad \sum_{1 \leq j < k \leq 5} v_{i_j} \cdot v_{i_k} \geq -2.$$

Although we do not consider here clauses of size larger than 4, we remark that for any k , and for any permutation $i_0, i_1, \dots, i_k$ on $0, 1, \dots, k$, $z_{12\dots k} \leq ((k+1) - \sum_{j=0}^k v_{i_j} \cdot v_{i_{j+1}})/4$, where the index $j+1$ is interpreted modulo $k+1$, is a valid constraint in the semidefinite programming relaxation of MAX k -SAT.

It is not difficult to verify that all these constraints are satisfied by any valid ‘integral’ assignment to the vectors v_i and the scalars z_i, z_{ij}, z_{ijk} and z_{ijkl} , i.e., an assignment in which $v_i = (1, 0, \dots, 0)$ if $x_i = 1$, and $v_i = (-1, 0, \dots, 0)$ if $x_i = 0$, and in which every z variable is set to 1 if its corresponding clause is satisfied by the assignment $x_1, x_2, \dots, x_n$, and to 0 otherwise. Thus, the presented semidefinite program is indeed a relaxation of the MAX 4-SAT instance.

The constraints of the above semidefinite program correspond to the *facets* of a polyhedron corresponding to the Boolean function $x_1 \vee x_2 \vee x_3 \vee x_4$. As explained in [17], it is therefore the strongest semidefinite relaxation that considers the clauses of the instance one by one. Stronger relaxations may be obtained by considering several clauses at once.

The semidefinite programming relaxation of a MAX 4-SAT instance has $n+1$ unknown unit vectors $v_0, v_1, \dots, v_n$ (the vectors v_{n+i} are just used as shorthands for $-v_i$), $O(n^4)$ scalar variables and $O(n^5)$ constraints. An almost optimal solution in which all unit vectors $v_0, v_1, \dots, v_n$ lie in $\mathbb{R}^{n+1}$ can be found in polynomial time ([1],[13],[19]).

3 Rounding Procedures

In this section we consider various procedures that can be used to round solutions of semidefinite programming relaxations and examine the performance guarantees that we get for MAX 4-SAT using them. We start with simple rounding procedures and then move on to more complicated ones. The new family of rounding procedures is then presented in Section 3.5.

3.1 Rounding Using a Random Hyperplane

Following Goemans and Williamson [12], many semidefinite programming based approximation algorithms round the solution to the semidefinite program using a *random hyperplane* passing through the origin. A random hyperplane that passes through the origin is chosen by choosing its normal vector r as a uniformly distributed vector on the unit sphere (in $\mathbb{R}^{n+1}$). A vector v_i is then rounded to 0 if v_i and v_0 fall on the same side of the random hyperplane, i.e., if $\text{sgn}(r \cdot v_i) = \text{sgn}(r \cdot v_0)$, and to 1, otherwise. Note that the rounded values of the variables are usually not independent. More specifically, if v_i and v_j are not perpendicular, i.e., if $v_i \cdot v_j \neq 0$, then the rounded values of x_i and x_j are dependent.

Given a set of unit vector V , we let $\text{prob}_H(V)$ denote the probability that not all the vectors of V fall on the same side of a random hyperplane that passes through the origin. It is not difficult to see that

$$\text{prob}_H(v_0, v_1) = \frac{\theta_{01}}{\pi} \quad , \quad \text{prob}_H(v_0, v_1, v_2) = \frac{\theta_{01} + \theta_{02} + \theta_{12}}{2\pi} \quad ,$$

where $\theta_{ij} = \arccos(v_i \cdot v_j)$ is the angle between v_i and v_j . Evaluating $\text{prob}_H(v_0, v_1, v_2, v_3)$ is more difficult. As noted in [17],

$$\text{prob}_H(v_0, v_1, v_2, v_3) = 1 - \frac{\text{Vol}(\lambda_{01}, \lambda_{02}, \lambda_{12}, \lambda_{03}, \lambda_{13}, \lambda_{23})}{\pi^2} \quad ,$$

where $(\lambda_{01}, \lambda_{02}, \lambda_{12}, \lambda_{03}, \lambda_{13}, \lambda_{23}) = \pi - (\theta_{23}, \theta_{13}, \theta_{03}, \theta_{12}, \theta_{02}, \theta_{01})$ and $\text{Vol}(\lambda_{01}, \lambda_{02}, \lambda_{12}, \lambda_{03}, \lambda_{13}, \lambda_{23})$ is the volume of a spherical tetrahedron with dihedral angles $\lambda_{01}, \lambda_{02}, \lambda_{12}, \lambda_{03}, \lambda_{13}, \lambda_{23}$. Unfortunately, the volume function of spherical tetrahedra seems to be a non-elementary function and numerical integration should be used to evaluate it ([24],[8],[15],[27]).

It is not difficult to verify, using inclusion-exclusion, that

$$\text{prob}_H(v_0, v_1, v_2, v_3, v_4) = \frac{1}{2} \sum_{i < j < k < l} \text{prob}_H(v_i, v_j, v_k, v_l) - \frac{1}{4} \sum_{i < j} \text{prob}_H(v_i, v_j) \quad .$$

Thus, the probability that certain five unit vectors are separated by a random hyperplane can be expressed as a combination of probabilities of events that involve at most four vectors and no further numerical integration is needed. More generally, $\text{prob}_H(v_0, \dots, v_k)$, for any *even* k , can be expressed as combinations of probabilities involving at most k vectors. The same does not hold, unfortunately, when k is odd.

We let $\text{relax}(v_0, v_1, \dots, v_i)$, where $1 \leq i \leq 4$, denote the ‘relaxed’ value of the clause $x_1 \vee \dots \vee x_i$, i.e., the maximal value to which the scalar $z_{12\dots i}$ can be set, while still satisfying all the relevant constraints, given the unit vectors $v_0, v_1, \dots, v_i$. We let

$$\text{ratio}_H(v_0, v_1, \dots, v_i) = \text{prob}_H(v_0, v_1, \dots, v_i) / \text{relax}(v_0, v_1, \dots, v_i) \quad .$$

Finally, for every $1 \leq i \leq 4$, we let $\alpha_i = \min \text{ratio}_H(v_0, v_1, \dots, v_i)$, where the minimum is over all configurations of unit vectors that satisfy the constraints

described in the previous section, and $\alpha = \min\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$. As follows from straightforward arguments (see [12] or [17]), α is a lower bound on the performance guarantee of the approximation algorithm for MAX 4-SAT that uses the semidefinite relaxation and then rounds the solution using a random hyperplane.

It is shown in [12] that $\alpha_1 = \alpha_2 \simeq 0.87856$. It is shown in [17] that $\alpha_3 = 7/8$. Unfortunately, it turns out that $\alpha_4 \simeq 0.845173$. The minimum is attained when the angle between each pair of vectors among $v_0, v_1, \dots, v_4$ is exactly $\arccos(1/5) \simeq 1.369438$. It is not difficult to check that when $v_i \cdot v_j = 1/5$, for every $0 \leq i < j \leq 4$, all the inequalities on z_{1234} simplify to $z_{1234} \leq 1$ and therefore $\text{relax}(v_0, v_1, v_2, v_3, v_4) = 1$.

3.2 Pre-rounding Rotations

Feige and Goemans [9] introduced the following variation of random hyperplane rounding. Let $f : [0, \pi] \rightarrow [0, \pi]$ be a continuous function satisfying $f(0) = 0$ and $f(\pi - \theta) = \pi - f(\theta)$, for $0 \leq \theta \leq \pi$. Before rounding the vectors using a random hyperplane, the vector v_i is rotated into a new vector v'_i , in the plane spanned by v_0 and v_i , so that the angle θ'_{0i} between v_0 and v'_i would be $\theta'_{0i} = f(\theta_{0i})$. The rotations of the vectors $v_1, \dots, v_n$ affects, of course, the angles between these vectors. Let θ'_{ij} be the angle between v'_i and v'_j . It is not difficult to see (see [9]), that for $i, j > 0$, $i \neq j$, we have

$$\cos \theta'_{ij} = \cos \theta'_{0i} \cdot \cos \theta'_{0j} + \frac{\cos \theta_{ij} - \cos \theta_{0i} \cdot \cos \theta_{0j}}{\sin \theta_{0i} \cdot \sin \theta_{0j}} \cdot \sin \theta'_{0i} \cdot \sin \theta'_{0j}.$$

The vectors $v_0, v'_1, \dots, v'_n$ are then rounded using a random hyperplane. The condition $f(0) = 0$ is required to ensure the continuity of the transformation $v_i \rightarrow v'_i$. The condition $f(\pi - \theta) = \pi - f(\theta)$ ensures that unnegated and negated literals are treated in the same manner.

Feige and Goemans [9] use rotations to obtain a 0.931-approximation algorithm for MAX 2-SAT. Rotations are also used in [29] and [30]. Can we use rotations to get a better approximation algorithm for MAX 4-SAT? The answer is that rotations on their own help, but very little. Consider the configuration v_0, v_1, v_2, v_3, v_4 in which $\theta_{0i} = \pi/2$, for $1 \leq i \leq 4$, and $\theta_{ij} = \arccos(1/3)$, for $1 \leq i < j \leq 4$. For this configuration we get $\text{relax} = 1$ and $\text{ratio}_H \simeq 0.8503$. As every rotation function f must satisfy $f(\pi/2) = \pi/2$, rotations have no effect on this configuration.

A different type of rotations was recently used by Nesterov [20] and Zwick [31]. These *outer-rotations* are used in [31] to obtain some improved approximation algorithms for MAX SAT and MAX NAE-SAT. We were not able to use them, however, to improve the results that we get in this paper for MAX 4-SAT.

3.3 Rounding the Vectors Independently

The semidefinite programming relaxation of MAX 4-SAT that we are using here is stronger than the linear programming relaxation suggested by Goemans and

Williamson [11]. It is nonetheless interesting to consider an adaptation of the rounding procedures used in [11] to the present context. The rounding procedures of [11] are based on the *randomized rounding* technique of Raghavan and Thompson [23],[22].

Let $g : [0, \pi] \rightarrow [0, \pi]$ be a continuous function such that $g(\pi - \theta) = \pi - g(\theta)$, for $0 \leq \theta \leq \pi$. Note again that we must have $g(\pi/2) = \pi/2$. We do not require $g(0) = 0$ this time. The rounding procedure described here rounds each vector *independently*. The variable x_i is assigned the value 1 with probability $g(\theta_{0i})/\pi$, and the value 0 with the complementary probability. The probability $\text{prob}_I(v_0, v_1, \dots, v_i)$ that a clause $x_1 \vee \dots \vee x_i$ is satisfied is now

$$\text{prob}_I(v_0, v_1, \dots, v_i) = 1 - \prod_{j=1}^i \left(1 - \frac{g(\theta_{0j})}{\pi} \right).$$

Note that as each vector is rounded independently, the angles θ_{ij} , where $i, j > 0$, between the vectors, have no effect this time. It may be worthwhile to note that the choice $g(\theta) = \pi/2$, for every $0 \leq \theta \leq \pi$, corresponds to choosing the assignment to the variables $x_1, x_2, \dots, x_n$ uniformly at random, a ‘rounding procedure’ that yields a ratio of $7/8$ for clauses of size 3 and $15/16$ for clauses of size 4. Goemans and Williamson [11] describe several functions g using which a $3/4$ -approximation algorithm for MAX SAT may be obtained.

Independent rounding performs well for long clauses. It cannot yield a ratio larger than $3/4$, however, for clauses of size 2. To see this, consider the configuration v_0, v_1, v_2 in which $\theta_{01} = \theta_{02} = \pi/2$ and $\theta_{12} = \pi$. We have $\text{relax}(v_0, v_1, v_2) = 1$ and $\text{prob}_I(v_0, v_1, v_2) = 3/4$, for any function g .

3.4 Simple Combinations

We have seen that hyperplane rounding works well for short clauses and that independent rounding works well for long clauses. It is therefore natural to consider a combination of the two.

Perhaps the most natural combination of hyperplane rounding and independent rounding is the following. Let $0 \leq \epsilon \leq 1$. With probability $1 - \epsilon$ round the vectors using a random hyperplane. With probability ϵ choose a random assignment. It turns out that the best choice of ϵ here is $\epsilon \simeq 0.086553$. With this value of ϵ , we get $\alpha_1 = \alpha_2 = \alpha_4 \simeq 0.853150$ while $\alpha_3 = 7/8$. Thus, we again get a small improvement but we are still far from $7/8$.

Instead of rounding all vectors using a random hyperplane, or choosing random values to all variables, we can round some of the vectors using a random hyperplane, and assign some of the variables random values. More precisely, we choose one random hyperplane. Each vector is now rounded using this random hyperplane with probability $1 - \epsilon$, or is assigned a random value with probability ϵ . The decisions for the different variables made independently. Letting $\epsilon \simeq 0.073609$, we get $\alpha_1 = \alpha_2 = \alpha_4 \simeq 0.856994$, while $\alpha_3 \simeq 0.874496$. This is again slightly better but still far from $7/8$.

3.5 More Complicated Combinations

Simple combinations of hyperplane rounding and independent rounding yield modest improvements. Can we get more substantial improvements by using more sophisticated combinations? To answer this question we introduce the following family of rounding procedures. The new family seems to include all the natural combinations of the rounding procedures mentioned above.

Each rounding procedure in the new family is characterized by three continuous functions $f, g : [0, \pi] \rightarrow [0, \pi]$ and $\epsilon : [0, \pi] \rightarrow [0, 1]$. The function f is used for rotating the vectors before rounding them using a random hyperplane, as described in Section 3.2. The function g is used to round the vectors independently, as described in Section 3.3. The function ϵ is used to decide which of the two roundings should be used. The decision is made *independently* for each vector, depending on the angle between it and v_0 . The function $\epsilon : [0, \pi] \rightarrow [0, 1]$ is a continuous function satisfying $\epsilon(\pi - \theta) = \epsilon(\theta)$, a condition that ensures that negated and unnegated literals are treated in the same manner. The vector v_i is rounded using a random hyperplane, shared by all the vectors rounded using a random hyperplane, with probability $1 - \epsilon(\theta_{0i})$, and is rounded independently, with probability $\epsilon(\theta_{0i})$. Vectors rounded using the shared hyperplane are rotated before the rounding. Let $v'_1, v'_2, \dots, v'_n$ be the vectors obtained by rotating the vectors $v_1, v_2, \dots, v_n$, as specified by the rotation function f . The probability that a clause $x_1 \vee x_2 \vee \dots \vee x_i$ is satisfied by the assignment produced by this combined rounding procedure is given by the following expression:

$$\text{prob}_C(v_0, v_1, \dots, v_i) = 1 - \sum_S \text{pr}(S) \cdot (1 - \text{prob}_H(v'(S))) \cdot (1 - \text{prob}_I(v(\bar{S})))$$

where

$$\text{pr}(S) = \prod_{i \in S} (1 - \epsilon(\theta_{0i})) \cdot \prod_{i \notin S} \epsilon(\theta_{0i}) ,$$

$$v'(S) = \{v_0\} \cup \{v'_i \mid i \in S\} \quad , \quad v(\bar{S}) = \{v_0\} \cup \{v_i \mid i \notin S\} ,$$

and where S ranges over all subsets of $\{1, 2, \dots, i\}$. Recall that $\text{prob}_H(u_1, u_2, \dots, u_k)$ is the probability that the set of vectors $u_1, u_2, \dots, u_k$ is separated by a random hyperplane, and that $\text{prob}_I(v_0, u_1, \dots, u_k)$ is the probability that at least one of the vectors $u_1, u_2, \dots, u_k$ is assigned the value 1 when all these vectors are rounded independently using the function g .

We have made some experiments with an even wider family of rounding procedures but we were not able to improve on the results obtained using rounding procedures selected from the family described here. More details will be given in the full version of the paper.

Can we select a rounding procedure from the proposed family of rounding procedures using which we can get an optimal, or an almost optimal, approximation algorithm for MAX 4-SAT?

4 The Search for Good Rounding Procedures

The new family of rounding procedures defined in the previous section is huge. How can we expect to select the best, or almost the best, rounding procedure from this family? As it turns out, although each rounding procedure is defined by three continuous functions f, g and ϵ , most of the values of these functions do not matter much. What really matter are the values of these functions at several ‘important’ angles. We therefore restrict ourselves to rounding procedures defined by *piecewise linear* functions f, g and ϵ with a relatively small number of bends. By placing these bends at the ‘important’ angles, we can find, as we shall see, a rounding procedure which is close to being the best rounding procedure from this family.

More specifically, we consider functions obtained by connecting k given points $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$ by straight line segments, where $x_1 = 0$ and $x_k = \pi/2$. For f we also require $y_1 = 0$ and $y_k = \pi/2$. For g we also require $y_k = \pi/2$. The values of the functions f, g and ϵ for $\pi/2 < \theta \leq \pi$ are determined by the conditions $f(\pi - \theta) = \pi - f(\theta)$, $g(\pi - \theta) = \pi - g(\theta)$ and $\epsilon(\pi - \theta) = \epsilon(\theta)$. We usually worked with $k \leq 5$.

For a given value of k we are now faced with a very difficult optimization problem in $6k - 9$ real variables, the variables being the x and y coordinates of the points through which the functions f, g and ϵ are required to pass. The objective is to maximize $\alpha(C(f, g, \epsilon))$, the performance guarantee obtained by using the rounding procedure defined by the functions f, g and ϵ that pass through the points. Recall that evaluating $\alpha(C(f, g, \epsilon))$ for a given set of functions f, g and ϵ is already a difficult task that requires finding the global minimum of a rather complicated function of 10 real variables.

We have written a Matlab program, called `opt_fun`, that tries to find a close to optimal rounding procedure that uses functions specified using at most k points. This is quite a non-trivial task and, as mentioned in the introduction, it required about 3000 lines of code, in addition to the sophisticated numerical optimization routines of Matlab’s optimization toolbox.

Although numerical methods were used to evaluate the performance guarantees of the different rounding procedures, we believe that the 0.8721 and $7/8$ performances ratio claimed for the two rounding procedures that will be described shortly are the correct performance ratios. There is, in fact, a completely mechanical way of generating a (long and tedious) rigorous proof of these claims. As mentioned in the introduction, we believe that it would be more fruitful to look for an algorithm that seems to achieve a performance ratio of $7/8$ before taking on the task of producing rigorous proofs. We believe that the use of numerical methods would be inevitable in the search for optimal algorithms for MAX 4-SAT and MAX SAT, at least using the current techniques.

5 Almost Optimal or Optimal Approximation Algorithms

We now present some optimal or close to optimal approximation algorithms obtained using rounding procedures from the new family of rounding procedures.

f	g	ϵ
(0 , 0)	(0 , 0)	(0 , 0.250000)
(0.777843 , 1.210627)	(0.750000 , 0)	(0.744611 , 0.357201)
(1.038994 , 1.445975)	(1.072646 , 0)	(1.039987 , 0.255183)
(1.248362 , 1.394099)	(1.248697 , 0.872552)	(1.072689 , 0.222928)
($\pi/2$, $\pi/2$)	($\pi/2$, $\pi/2$)	($\pi/2$, 0.131681)

Fig. 1. The rounding procedure that seems to yield a 0.8721-approximation algorithm for MAX 4-SAT

f	g	ϵ
(0 , 0)	(0 , 0.550000)	(0 , 0.650000)
(1.394245 , 1.544705)	(1.155432 , 1.154866)	(0.413021 , 0.163085)
($\pi/2$, $\pi/2$)	(1.394111 , 0.931661)	($\pi/2$, 0.160924)
	($\pi/2$, $\pi/2$)	

Fig. 2. The rounding procedure that seems to yield an optimal 7/8-approximation algorithm for MAX {2,3,4}-SAT.

5.1 MAX 4-SAT

Using the semidefinite programming relaxation of Section 2 and the rounding procedure defined by the three piecewise linear functions passing through the points given in Figure 1 we seem to obtain a 0.8721-approximation algorithm for MAX 4-SAT, or more specifically, an algorithm with $\alpha_1 \simeq \alpha_2 \simeq \alpha_3 \simeq \alpha_4 \simeq 0.8721$. As we shall see in Section 6, this is essentially the best approximation ratio that we can obtain using a rounding procedure from the family considered.

It is interesting to note that $g(\theta) = 0$ for $0 \leq \theta \leq 1.072646$ and that $0.13 \leq \epsilon(\theta) \leq 0.36$ for $0 \leq \theta \leq \pi$. This means that if the angle θ_{0i} between v_i and v_0 is less than about $\pi/3$, then with a probability of about $1/4$, the variable x_i is assigned the value 0, without any further consideration of the angle θ_{0i} . It is also interesting to note that the function $f(\theta)$ is not monotone.

5.2 MAX {2,3,4}-SAT

Using the semidefinite programming relaxation of Section 2 and the rounding procedure defined by the three piecewise linear functions passing through the points given in Figure 2 we believe we obtain a 7/8-approximation algorithm for MAX {2,3,4}-SAT. We get in fact, an approximation algorithm for MAX 4-SAT with $\alpha_2 \simeq 0.8751$, $\alpha_3 = 7/8$, $\alpha_4 \simeq 0.8755$ but with $\alpha_1 \simeq 0.8352$. It is interesting

to note the non-monotonicity of the function $g(\theta)$ and the fact that only one intermediate point is needed for $f(\theta)$ and $\epsilon(\theta)$ and only two intermediate points are needed for $g(\theta)$.

A $7/8$ -approximation algorithm for MAX $\{2, 3, 4\}$ -SAT is of course optimal as a ratio better than $7/8$ cannot be obtained even for MAX $\{3\}$ -SAT, which is a subproblem of MAX $\{2, 3, 4\}$ -SAT.

5.3 MAX 3-SAT

The optimal $7/8$ -approximation algorithm for MAX 3-SAT presented in [17] has $\alpha_1 = \alpha_2 \simeq 0.87856$ and $\alpha_3 = 7/8$. Using pre-rounding rotations we can obtain an approximation algorithm for MAX 3-SAT with $\alpha_1 = \alpha_2 \simeq 0.9197$ and $\alpha_3 = 7/8$. This algorithm would perform better than the algorithm of [17] on instances in which some of the contribution to the optimal value of their semidefinite programming relaxation comes from clauses of size one or two. The details of this algorithm will be given in the full version of the paper.

5.4 MAX 2-SAT

Feige and Goemans [9] obtained an approximation algorithm for MAX 2-SAT with $\alpha_1 \simeq 0.976$ and $\alpha_2 \simeq 0.931$. Although we cannot improve α_2 , the performance ratio on clauses of size two, we can obtain, using pre-rounding rotations, an approximation algorithm for MAX 2-SAT with $\alpha_1 \simeq 0.983$ and $\alpha_2 \simeq 0.931$. The details of this algorithm will be given in the full version of the paper.

6 Limitations of Current Rounding Procedures

We presented above a rounding procedure using which we seem to get a 0.8721 -approximation algorithm for MAX 4-SAT. This is extremely close to $7/8$. Could it be that by searching a little bit harder, or perhaps allowing more bends, we could find a rounding procedure from the family defined in Section 3.5 using which we could obtain an optimal $7/8$ -approximation algorithm for MAX 4-SAT? Unfortunately, the answer is no. We show in this section that the rounding procedure described in Section 5.1 is close to being the best rounding procedure of the family considered.

Let θ_{ij} , for $0 \leq i < j \leq 4$, be the angles between the five unit vectors v_0, v_1, v_2, v_3, v_4 . Let $c_{ij} = \cos \theta_{ij}$. It is not difficult to check that if

$$\begin{aligned} c_{12} &= \frac{1+c_{01}+c_{02}-2c_{03}-2c_{04}}{3} & , & \quad c_{13} = \frac{1+c_{01}-2c_{02}+c_{03}-2c_{04}}{3} \\ c_{23} &= \frac{1-2c_{01}+c_{02}+c_{03}-2c_{04}}{3} & , & \quad c_{14} = \frac{1+c_{01}-2c_{02}-2c_{03}+c_{04}}{3} \\ c_{24} &= \frac{1-2c_{01}+c_{02}-2c_{03}+c_{04}}{3} & , & \quad c_{34} = \frac{1-2c_{01}-2c_{02}+c_{03}+c_{04}}{3} \end{aligned}$$

then $\text{relax}(v_0, v_1, v_2, v_3, v_4) = 1$.

Let $0 < \theta_1 < \theta_2 \leq \pi/2$ be two angles. Consider the configuration (v_0, v_1) in which $\theta_{01} = \pi - \theta_1$, and the two configurations $(v_0, v_1^1, v_2^1, v_3^1, v_4^1)$ and $(v_0, v_1^2, v_2^2, v_3^2, v_4^2)$ in which

$$(\theta_{01}^1, \theta_{02}^1, \theta_{03}^1, \theta_{04}^1) = (\theta_1, \theta_1, \theta_1, \pi - \theta_2) \quad , \quad (\theta_{01}^2, \theta_{02}^2, \theta_{03}^2, \theta_{04}^2) = (\theta_2, \theta_2, \theta_2, \theta_2)$$

and in which the angles θ_{jk}^i , for $1 \leq j < k \leq 4$ are determined according to the relations above so that $\text{relax}(v_0, v_1^i, v_2^i, v_3^i, v_4^i) = 1$, for $i = 1, 2$. It is not difficult to check that $\theta_{12}^1 = \theta_{13}^1 = \theta_{23}^1 = \arccos(\frac{1+2\cos\theta_2}{3})$, $\theta_{14}^1 = \theta_{24}^1 = \theta_{34}^1 = \arccos(\frac{1-3\cos\theta_1-\cos\theta_2}{3})$ and that $\theta_{ij}^2 = \arccos(\frac{1-2\cos\theta_2}{3})$, for $1 \leq j < k \leq 4$. Assume that the configurations $(v_0, v_1^i, v_2^i, v_3^i, v_4^i)$, for $i = 1, 2$, are feasible. For every rounding procedure C we have

$$\alpha(C) \leq \min\{ \text{ratio}_C(v_0, v_1), \text{ratio}_C(v_0, v_1^1, v_2^1, v_3^1, v_4^1), \text{ratio}_C(v_0, v_1^2, v_2^2, v_3^2, v_4^2) \}.$$

As the only angles between v_0 and other vectors in these three configurations are $\theta_1, \theta_2, \pi - \theta_1$ and $\pi - \theta_2$, and as $f(\pi - \theta) = \pi - f(\theta)$, $g(\pi - \theta) = \pi - g(\theta)$ and $\epsilon(\pi - \theta) = \epsilon(\theta)$, we get that for every rounding procedure from our family, the three ratios $\text{ratio}_C(v_0, v_1)$, $\text{ratio}_C(v_0, v_1^1, v_2^1, v_3^1, v_4^1)$ and $\text{ratio}_C(v_0, v_1^2, v_2^2, v_3^2, v_4^2)$ depend only on the six parameters $f(\theta_1), f(\theta_2), g(\theta_1), g(\theta_2), \epsilon(\theta_1)$ and $\epsilon(\theta_2)$.

Take $\theta_1 = 0.95$ and $\theta_2 = \arccos(1/5) \simeq 1.369438$. It is possible to check that the resulting two configurations $(v_0, v_1^1, v_2^1, v_3^1, v_4^1)$ and $(v_0, v_1^2, v_2^2, v_3^2, v_4^2)$ are feasible. The choice of the six parameters that maximizes the minimum ratio of the three configurations, found again using numerical optimization, is:

$$\begin{aligned} f(\theta_1) &\simeq 1.410756 & , & \quad g(\theta_1) \simeq 0 & , & \quad \epsilon(\theta_1) \simeq 0.309376 \\ f(\theta_2) &\simeq 1.448494 & , & \quad g(\theta_2) \simeq 1.233821 & , & \quad \epsilon(\theta_2) \simeq 0.122906 \end{aligned}$$

With this choice of parameters, the three ratios evaluate to about 0.8724. No rounding procedure from the family can therefore attain a ratio of more than 0.8724 simultaneously on these three specific configurations. No rounding procedure from the family can therefore yield a performance ratio greater than 0.8724 for MAX 4-SAT, even if the functions f, g and ϵ are not piecewise linear.

7 The Quality of the Semidefinite Programming Relaxation

Let I be an instance of MAX 4-SAT. Let $\text{opt}(I)$ be the value of the optimal assignment for this instance. Let $\text{opt}^*(I)$ be the value of the optimal solution of the canonical semidefinite programming relaxation of the instance given in Section 2. Clearly $\text{opt}(I) \leq \text{opt}^*(I)$ for every instance I . The *integrality ratio* of the relaxation is defined to be $\gamma = \inf_I \text{opt}(I)/\text{opt}^*(I)$, where the infimum is taken over all the instances.

In Section 3, when we analyzed the performance of different rounding procedures, we compared the value, or rather the expected value, of the assignment produced by a rounding procedure to $\text{opt}^*(I)$, the optimal value of the semidefinite programming relaxation. It is not difficult to see that any lower bound α on the performance ratio of a rounding procedure obtained in this way would satisfy $\alpha \leq \gamma$. Thus, the rounding procedure of Section 5.1 seems to imply that $\gamma \geq 0.8721$. In this section we describe *upper bounds* on the integrality ratio γ , thereby obtaining upper bounds on the performance ratios that can be obtained

by *any* approximation algorithm that uses the relaxation of Section 2, at least using the type of analysis used in Section 3.

It is shown in [17] that the integrality ratio of the canonical semidefinite programming relaxation of MAX 3-SAT is exactly $\gamma_3 = 7/8$. As the canonical relaxations of MAX 3-SAT and MAX 4-SAT coincide on instances of MAX 3-SAT, we get that $\gamma = \gamma_4 \leq 7/8$.

We can show, that the integrality ratio of the canonical relaxation of MAX 4-SAT, given in Section 2, is at most 0.8753, even when restricted to instances of MAX $\{1, 4\}$ -SAT, i.e., to instances of MAX 4-SAT in which all clauses are of size 1 or 4. Though this upper bound does not preclude the possibility of obtaining an optimal $7/8$ -approximation algorithm for MAX 4-SAT using the canonical semidefinite programming relaxation of the problem, the closeness of this upper bound to $7/8$ does indicate that it will not be easy, even if clauses of length 3 are not present. It may be necessary to consider stronger relaxations of MAX 4-SAT, e.g., relaxations obtained by considering several clauses of the instance at once.

8 Concluding Remarks

We have come frustratingly close to obtaining an optimal $7/8$ -approximation algorithm for MAX 4-SAT. We have seen that devising a $7/8$ -approximation algorithm for MAX $\{1, 4\}$ -SAT is already a challenging problem. Note that Håstad's $7/8$ upper bound for MAX 3-SAT and MAX 4-SAT does not apply to MAX $\{1, 4\}$ -SAT, as clauses of length three are not allowed in this problem. A gadget (see [26]) supplied by Greg Sorkin shows that no polynomial time approximation algorithm for MAX $\{1, 4\}$ -SAT can have a performance ratio greater than $9/10$, unless $P=NP$.

We believe that optimal $7/8$ -approximation algorithms for MAX 4-SAT and MAX SAT do exist. The fact that we have come so close to obtaining such algorithms may in fact be seen as cause for optimism. There is still a possibility that simple extensions of ideas laid out here could be used to achieve this goal. If this fails, it may be necessary to attack the problems from a more global point of view. Note that the analysis carried out here was very local in nature. We only considered one clause of the instance at a time. As a result we only obtained lower bounds on the performance ratios of the algorithms considered. It may even be the case that the algorithms from the family of algorithms considered here do give a performance ratio of $7/8$ for MAX 4-SAT although a more global analysis is required to show it.

We also hope that MAX 4-SAT would turn out to be the last barrier on the road to an optimal approximation algorithm for MAX SAT. The almost optimal algorithms for MAX 4-SAT presented here may be used to obtain an almost optimal algorithm for MAX SAT. We have not worked out yet the exact bounds that we can get for MAX SAT as we still hope to get an optimal algorithm for MAX 4-SAT before proceeding with MAX SAT.

Finally, a word on our methodology. Our work is a bit unusual as we use experimental and numerical means to obtain *theoretical* results. We think that

the nature of the problems that we are trying to solve calls for this approach. No one can rule out, of course, the possibility that some clever new ideas would dispense with most of the technical difficulties that we are facing here. Until that happens, however, we see no alternative to the current techniques. The use of experimental and numerical means does not mean that we have to give up the rigorosity of the results. Once we obtain the ‘right’ result, we can devote efforts to proving it rigorously, possibly using automated means.

References

1. F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1995.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45:501–555, 1998.
3. S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45:70–122, 1998.
4. T. Asano. Approximation algorithms for MAX SAT: Yannakakis vs. Goemans-Williamson. In *Proceedings of the 3rd Israel Symposium on Theory and Computing Systems, Ramat Gan, Israel*, pages 24–37, 1997.
5. T. Asano, T. Ono, and T. Hirata. Approximation algorithms for the maximum satisfiability problem. *Nordic Journal of Computing*, 3:388–404, 1996.
6. M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27:804–915, 1998.
7. M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing, San Diego, California*, pages 294–304, 1993. See Errata in STOC’94.
8. H.S.M. Coxeter. The functions of Schläfli and Lobatschewsky. *Quarterly Journal of Mathematics (Oxford)*, 6:13–29, 1935.
9. U. Feige and M.X. Goemans. Approximating the value of two prover proof systems, with applications to MAX-2SAT and MAX-DICUT. In *Proceedings of the 3rd Israel Symposium on Theory and Computing Systems, Tel Aviv, Israel*, pages 182–189, 1995.
10. U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43:268–292, 1996.
11. M.X. Goemans and D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.
12. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.
13. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, 1993. Second corrected edition.
14. J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, Texas*, pages 1–10, 1997. Full version available as E-CCC Report number TR97-037.
15. W.Y. Hsiang. On infinitesimal symmetrization and volume formula for spherical or hyperbolic tetrahedrons. *Quarterly Journal of Mathematics (Oxford)*, 39:463–468, 1988.

16. D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
17. H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38rd Annual IEEE Symposium on Foundations of Computer Science, Miami Beach, Florida*, pages 406–415, 1997.
18. S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proceedings of the 35rd Annual IEEE Symposium on Foundations of Computer Science, Santa Fe, New Mexico*, pages 819–830, 1994.
19. Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming*. SIAM, 1994.
20. Y. E. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9:141–160, 1998.
21. C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.
22. P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.
23. P. Raghavan and C. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
24. L. Schläfli. On the multiple integral $\int^n dx dy \dots dz$, whose limits are $p_1 = a_1x + b_1y + \dots + h_1z > 0, p_2 > 0, \dots, p_n > 0$, and $x^2 + y^2 + \dots + z^2 < 1$. *Quarterly Journal of Mathematics (Oxford)*, 2:269–300, 1858. Continued in Vol. 3 (1860), pp. 54–68 and pp. 97–108.
25. L. Trevisan. Approximating satisfiable satisfiability problems. In *Proceedings of the 5th European Symposium on Algorithms, Graz, Austria*, 1997. 472–485.
26. L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming (extended abstract). In *Proceedings of the 37rd Annual IEEE Symposium on Foundations of Computer Science, Burlington, Vermont*, pages 617–626, 1996.
27. E.B. Vinberg. Volumes of non-Euclidean polyhedra. *Russian Math. Surveys*, 48:15–45, 1993.
28. M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.
29. U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, California*, pages 201–210, 1998.
30. U. Zwick. Finding almost-satisfying assignments. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, Texas*, pages 551–560, 1998.
31. U. Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *Proceedings of the 31th Annual ACM Symposium on Theory of Computing, Atlanta, Georgia*, 1999. To appear.

On the Chvátal Rank of Certain Inequalities

Mark Hartmann¹, Maurice Queyranne^{2,*}, and Yaoguang Wang³

¹ University of North Carolina, Chapel Hill, N.C., U.S.A. 27599
mudville@imap.unc.edu

² University of British Columbia, Vancouver, B.C., Canada V6T 1Z2
Maurice.Queyranne@commerce.ubc.ca

³ PeopleSoft Inc., San Mateo, CA, U.S.A. 94404
Yaoguang_Wang@peoplesoft.com

Abstract. The Chvátal rank of an inequality $ax \leq b$ with integral components and valid for the integral hull of a polyhedron P , is the minimum number of rounds of Gomory-Chvátal cutting planes needed to obtain the given inequality. The Chvátal rank is at most one if b is the integral part of the optimum value $z(a)$ of the linear program $\max\{ax : x \in P\}$. We show that, contrary to what was stated or implied by other authors, the converse to the latter statement, namely, the Chvátal rank is at least two if b is less than the integral part of $z(a)$, is not true in general. We establish simple conditions for which this implication is valid, and apply these conditions to several classes of facet-inducing inequalities for travelling salesman polytopes.

1 Introduction

We consider the problem of deciding whether the Chvátal rank of an inequality, relative to a given polyhedron, is greater than one or not. Formally, given a finite set E , we let $\mathbb{R}^E$ denote the space of vectors with real-valued components indexed by E , and $\mathbb{Z}^E$ the set of all vectors in $\mathbb{R}^E$ all of whose components are integer. The integral part $\lfloor y \rfloor$ of a real number y is the largest integer less than or equal to y . Given a polyhedron P in $\mathbb{R}^E$, let $P_I \doteq \text{conv}(P \cap \mathbb{Z}^E)$ be its *integral hull*, that is, the convex hull of the integer points in P . Following Chvátal et al. [11], let

$$P' \doteq \left\{ x \in P : ax \leq a_0 \text{ whenever } a \in \mathbb{Z}^E, a_0 \in \mathbb{Z} \text{ and } \max\{ax : x \in P\} < a_0 + 1 \right\}.$$

So $P_I \subseteq P' \subseteq P$, and we may think of P' as being derived from P by a single round of adding cutting planes $ax \leq a_0 \doteq \lfloor b \rfloor$ derived from any valid inequality $ax \leq b$ for P with integral left hand side vector a . Defining $P^{(0)} \doteq P$ and $P^{(j)} \doteq (P^{(j-1)})'$ for all positive integers j , Schrijver [25] shows that, when P is

* Research supported by grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada

a rational polyhedron, each $P^{(j)}$ thus defined is itself a rational polyhedron and there exists a nonnegative integer r such that $P^{(r)} = P_I$. (Chvátal [10] shows this when P is bounded.) Consider any inequality $ax \leq a_0$ with $a \in \mathbb{Z}^E$ and $a_0 \in \mathbb{Z}$, which is valid for P_I . Its *Chvátal rank* $r_P(a, a_0)$ is the smallest j such that $ax \leq a_0$ is valid for $P^{(j)}$. Thus $r_P(a, a_0) = 0$ iff the inequality $ax \leq a_0$ is valid for P ; $r_P(a, a_0) = 1$ iff it is valid for P' but not for P ; and $r_P(a, a_0) \geq 2$ iff it is valid for P_I but not for P' .

The Chvátal rank of an inequality is often considered as a measure of its “complexity” relative to the formulation defining polyhedron P . Rank-one inequalities may be derived by a generally straightforward rounding argument. In contrast, inequalities of rank two or greater require a more complicated proof, involving several stages of rounding. In this paper, we consider the problem of determining whether or not the Chvátal rank a given inequality is at least two, as considered, among others, in [2], [18] and [4], [5], [6], [16] and [20] for several classes of inequalities for travelling salesman polytopes. See also Chvátal et al. [11] for references to a number of results and conjectures regarding lower bounds on the Chvátal rank of inequalities and polyhedra related to various combinatorial optimization problems and more recently Bockmayr et al. [3] and Eisenbrand and Schulz [15] for polynomial upper bounds on the Chvátal rank of 0–1 polytopes. See also Caprara and Fischetti [7], Caprara et al. [8] and Eisenbrand [14] regarding the separation problem for inequalities with Chvátal rank one.

A related notion of *disjunctive rank* was introduced in the seminal work of Balas et al. [1]; see also [9]. Their lift-and-project approach, which does not rely on rounding, applies to polyhedra whose integer variables are restricted to be 0-1 valued. It does not apply (directly) to general integer variables; it is, in this sense, less general than the Gomory-Chvátal approach. On the other hand, it applies directly to *mixed* 0-1 programs, thus allowing continuous variables. It also leads to interesting mathematical and computational properties. We outline below some of those properties that are most directly related to the present paper. Let $K \subseteq \mathbb{R}^N$ be a polyhedron and $E \subseteq N$ be the set of 0-1 variables. For $S \subseteq E$ and any assignment $\bar{x}_S \in \{0, 1\}^S$ of 0-1 values to the variables in S , let $K_S(\bar{x}_S) = K \cap \{x \in \mathbb{R}^N : x_j = \bar{x}_j \forall j \in S\}$. Let $K_S = \bigcup \{K_S(\bar{x}_S) : \bar{x}_S \in \{0, 1\}^S\}$, so we have $K = K_\emptyset \supseteq K_S \supseteq K_E$, and K_E is the set of all feasible mixed 0-1 solutions associated with K and E . An inequality $ax \leq a_0$, valid for K_E , has *disjunctive rank* r if r is the smallest integer such that there exists a subset S of E with $|S| = r$ and $ax \leq a_0$ valid for K_S . It follows immediately that the disjunctive rank of any such inequality is at most $|E|$, the number of 0-1 variables. This is in sharp contrast with the fact that the Chvátal rank of certain inequalities can be unbounded, even for polytopes with two variables and integral hull contained in $\{0, 1\}^2$ (e.g., [23], page 227). Eisenbrand and Schulz [15] show that, if $P = K$ is contained in the unit cube, then the Chvátal rank of any inequality, valid for its integral hull $P_I = K_N$, is $O(|N|^2 \log |N|)$; this upper bound is still much larger than the corresponding bound of $|N|$ for the disjunctive rank. Balas et al. also show that the disjunctive and Chvátal ranks of

an inequality are “incomparable”; that is, for certain inequalities, the disjunctive rank is larger than their Chvátal rank, whereas the converse holds for other inequalities. An important computational property of the disjunctive rank is that, for any *fixed* r , it can be decided in polynomial time whether or not the disjunctive rank of a given inequality $ax \leq a_0$ is at most r . Indeed, enumerate all $\binom{|E|}{r} = O(|E|^r)$ subsets $S \subseteq E$ with $|S| = r$; for any such S , the inequality is valid for K_S if and only if it is valid for all 2^r polyhedra $K_S(\bar{x}_S)$ with $\bar{x}_S \in \{0, 1\}^S$ (if $K_S(\bar{x}_S) = \emptyset$ then the inequality is trivially valid); since r is fixed, validity for K_S can thus be verified by solving a fixed number 2^r of linear programs $\max\{ax : x \in K_S(\bar{x}_S)\}$, each about the same size as the linear system defining K ; this yields the desired polynomial time algorithm. In contrast, it follows from a result of Eisenbrand [14] and the equivalence between separation and optimization (e.g., [23]), that it is NP-hard to decide whether or not the Chvátal rank of an inequality is *at most one*. In this paper we present *sufficient* conditions for a negative answer to this decision problem.

For $a \in \mathbb{Z}^E$ and $a_0 \in \mathbb{Z}$, a straightforward method for proving that the Chvátal rank $r_P(a, a_0)$ is zero or one, relies on the optimal value $z_P(a)$ of the linear programming problem

$$z_P(a) \doteq \max\{ax : x \in P\}$$

as follows. If $a_0 \geq z_P(a)$ then $r_P(a, a_0) = 0$. If $\lfloor z_P(a) \rfloor = a_0 < z_P(a)$ then $r_P(a, a_0) = 1$. A number of researchers have inferred from these elementary observations that the following implication must hold:

$$a_0 < \lfloor z_P(a) \rfloor \implies r_P(a, a_0) \geq 2. \quad (1)$$

This is stated explicitly in Lemma 3.1 in [6] and is used in Theorem 3.3 therein to prove that the Chvátal rank of certain clique tree inequalities for the symmetric Travelling Salesman Problem (TSP) is at least two, where P is the corresponding subtour polytope (see Sect. 3 below for details), by Giles and Trotter [18] (p. 321) to disprove a conjecture of Edmonds that each non-trivial facet of the stable set polytope of a $K_{1,3}$ -free graph has Chvátal rank one, where P is described by the non-negativity and clique constraints, and by Balas and Saltzman in Proposition 3.8 in [2] to show that the Chvátal rank of inequalities induced by certain cliques of the complete tri-partite graph $K_{n,n,n}$ is at least (in fact, exactly) two, where P is the linear programming relaxation of the (axial) three-index assignment problem. This result is also implicit in related statements in [4] (p. 265) for the envelope inequality for the symmetric TSP, and in [16] (p. 263) for a class of T -lifted comb inequalities for the asymmetric TSP.

Example 1 below shows that implication (1) may fail if the inequality is not properly scaled.

Example 1: Let $P \doteq \{x = (x_1, x_2) \in \mathbb{R}^2 : 2x_1 + 3x_2 \leq 11, x \geq 0\}$. The inequality $ax \leq a_0$ with $a = (3, 3)$ and $a_0 = 15$ is facet-inducing for $P_I = \text{conv}\{(0, 0), (0, 3), (1, 3), (4, 1), (5, 0)\}$. We have $z_P(a) = a(5\frac{1}{2}, 0) = 16\frac{1}{2}$ so $a_0 < \lfloor z_P(a) \rfloor$. Yet $r_P(a, a_0) = 1$, since $3x_1 + 3x_2 \leq 15$ is equivalent to $x_1 + x_2 \leq 5$, which is valid for P' since $z_{P'}(1, 1) = 5\frac{1}{2}$. $\square$

We should therefore assume that the components of a are relatively prime integers. Example 2 below, however, shows that this assumption is still not sufficient for the validity of implication (1).

Example 2: With polyhedron P of Example 1, now let $a = (3, 1)$ and $a_0 = 15$. Inequality $ax \leq a_0$ is valid for P_I , with equality attained for $x = (5, 0)$. We have $z_P(a) = 16\frac{1}{2}$ so here again $a_0 < \lfloor z_P(a) \rfloor$. Yet $r_P(a, a_0) = 1$, since $3x_1 + x_2 \leq 15$ is implied by $x_1 + x_2 \leq 5$ and $x_1 \leq 5$ (the latter inequality is valid for P' since $z_P(1, 0) = 5\frac{1}{2}$). $\square$

Note that the inequality $3x_1 + x_2 \leq 15$ in Example 2, not being facet-inducing for P_I , is implied by other inequalities. This allowed us to obtain a large enough integrality gap $z_P(a) - \lfloor z_P(a) \rfloor$ while retaining relatively prime components. Because most inequalities of interest in combinatorial optimization are facet-inducing for the integral hull P_I , we shall restrict attention to these cases. Corollary 3 herein will show that the preceding two conditions on an inequality (that is, with relatively prime components, and facet-inducing for P_I) are sufficient for implication (1) to hold, when P_I is *full dimensional*.

Example 3 below, however, shows that this need not be the case when P_I is not full dimensional. (Another, more complicated example using a TSP facet-inducing inequality is given in [5].)

Example 3: Polyhedron $P \doteq \{x = (x_1, x_2) \in \mathbb{R}^2 : 2x_1 + 3x_2 = 11, x \geq 0\}$ is derived from that of Example 1 above by turning the first inequality constraint into an equation. The inequality $ax \leq a_0$ with $a = (1, 0)$ and $a_0 = 4$ has relatively prime components, and is facet-inducing for $P_I = \text{conv}\{(1, 3); (4, 1)\}$. We have $z_P(a) = a(5\frac{1}{2}, 0) = 5\frac{1}{2}$ so $a_0 < \lfloor z_P(a) \rfloor$. Yet $r_P(a, a_0) = 1$, since $x_1 \leq 4$ is implied by inequalities $-2x_1 - 3x_2 \leq -11$ and $x_1 + x_2 \leq 5$ which are valid for P' . $\square$

In this paper, we give simple sufficient conditions for implication (1) to hold. We then use these conditions to present correct proofs that the inequalities from [2], [4], [6] [16] and [18] cited above have Chvátal rank at least two, where P is the corresponding relaxation. Some of these conditions were stated without proof and used in [5] to prove that all ladder inequalities have Chvátal rank two for the symmetric TSP.

The content of the present paper is as follows. In Sect. 2, we define a sufficient condition, called reduced integral form, for implication (1) to hold. This result implies in particular that (1) holds if P_I is full dimensional and inequality $ax \leq a_0$ is facet-inducing for P_I and has relatively prime integer components. We also present a relatively simple sufficient condition for an inequality to be in reduced integral form. In Sections 3 and 4, we consider some classes of inequalities for the symmetric and asymmetric TSP, respectively. We show how to apply this sufficient condition and prove that several classes of TSP inequalities, including those referenced above, have Chvátal rank at least two relative to the subtour polytope.

2 Reduced Integral Forms and B -Canonical Forms

A system of linear equations $Cx = d$ represents the *equality system* for a polyhedron $P \subseteq \mathbb{R}^E$ iff all equations valid for all $x \in P$ are linear combinations of the equations in $Cx = d$, and if it is minimal for this property. Recall that all systems $Cx = d$ representing the equality system for P have the same number of equations, which we denote by $m(P)$. The dimension of P is $\dim(P) \doteq |E| - m(P)$. Other polyhedral concepts and results used below are found in [23].

An *integral polyhedron* is a rational polyhedron in $\mathbb{R}^E$ with at least one extreme point, and such that all its extreme points are integral. An inequality $ax \leq a_0$ is a *reduced integral form* for integral polyhedron Q if

- (i) it is facet-inducing for Q ;
- (ii) all its components a_e ($e \in E$) are integers, that is, $a \in \mathbb{Z}^E$; and
- (iii) there exists $z \in \mathbb{Z}^E$ with $az = 1$ and $Cz = 0$, where $Cx = d$ is any linear system representing the equality system for Q .

Remark. It follows immediately from condition (iii) that an inequality $ax \leq a_0$ in reduced integral form has relatively prime integral components. In this case, we can interpret the rounding down of the right-hand-side b to $\lfloor b \rfloor$ as moving the supporting hyperplane to P towards Q until it first hits an integral point. Condition (iii) ensures that at least one such point lies in the affine space of Q (this can be shown starting from an integer point in the facet of Q determined by $ax \leq a_0$).

Our first result relates reduced integral forms to condition (1), which is the nontrivial half of equivalence (2) below (see Theorem 2.2.3 of Hartmann [21]).

Theorem 1 *Let $ax \leq a_0$ be a reduced integral form for integral polyhedron Q . The equivalence*

$$r_P(a, a_0) \leq 1 \quad \text{iff} \quad a_0 \geq \lfloor z_P(a) \rfloor \quad (2)$$

holds for any rational polyhedron P such that $m(Q) = m(P)$ and $\text{conv}(P \cap \mathbb{Z}^E) = Q$.

Proof. Assume that $ax \leq a_0$ is a reduced integral form for Q , so that $a_0 \in \mathbb{Z}$ and $Q = \text{conv}(P \cap \mathbb{Z}^E)$. The implication $a_0 \geq \lfloor z_P(a) \rfloor \Rightarrow r_P(a, a_0) \leq 1$ follows directly from the definition of the Chvátal rank. To prove the converse implication, we assume that $r_P(a, a_0) \leq 1$ and aim to prove that $z_P(a) < a_0 + 1$. Since $ax \leq a_0$ is valid for P' and P' is a rational polyhedron, the inequality $ax \leq a_0$ is implied by the inequalities and equations in a minimal linear system defining P' . Since $Q \subseteq P' \subseteq P$ and $m(Q) = m(P)$, we may assume that $Cx = d$ is the equality system for P , P' and Q . Therefore, $a = \sum_k \mu_k c^k + \nu C$ and $a_0 \geq \sum_k \mu_k c_0^k + \nu d$, where each inequality $c^k x \leq c_0^k$ is facet-inducing for P' , has integral components and satisfies $z_P(c^k) < c_0^k + 1$; all $\mu_k \in \mathbb{R}$ satisfy $\mu_k \geq 0$; and $\nu \in \mathbb{R}^{m(Q)}$. Since $ax \leq a_0$ is facet-inducing for Q , and all inequalities $c^k x \leq c_0^k$ are valid for Q , it follows that at least one of these inequalities, say, $c^1 x \leq c_0^1$, induces the same facet of Q as $ax \leq a_0$. Therefore, there exist a vector

$\lambda \in \mathbb{R}^{m(Q)}$ and a scalar $s > 0$ such that $(a, a_0) = s(c^1, c_0^1) + \lambda(C, d)$. If z is the “certificate” from condition (iii), then

$$1 = az = (sc^1 + \lambda C)z = s(c^1 z) + \lambda(Cz) = s(c^1 z),$$

which implies that $s \leq 1$ since $c^1 z$ is a (positive) integer. Since $z_P(c^1) < c_0^1 + 1$ and $0 < s \leq 1$, we have

$$z_P(a) = z_P(sc^1 + \lambda C) = sz_P(c^1) + \lambda d < s(c_0^1 + 1) + \lambda d = a_0 + s \leq a_0 + 1 .$$

The proof is complete. $\square$

Example 4: This example shows that we cannot drop the requirement that $m(P) = m(Q)$. Let

$$P \doteq \{x = (x_1, x_2) \in \mathbb{R}^2 : 4x_1 + 3x_2 \leq 4, x_1 + x_2 \geq 1, 4x_1 + x_2 \geq 0\} .$$

The inequality $ax \leq a_0$ with $a = (0, 1)$ and $a_0 = 1$ is facet-inducing for $P_I = \text{conv}\{(1, 0); (0, 1)\}$, whose equality set is $x_1 + x_2 = 1$, and is in reduced integral form as shown by the “certificate” $z = (-1, 1)$. We have $z_P(a) = 2$ so $a_0 < \lfloor z_P(a) \rfloor$. Yet $r_P(a, a_0) = 1$, since $x_2 \leq 1$ is implied by $x_1 + x_2 \leq 1$ and $-x_1 \leq 0$ (which are valid for P' since $z_P(1, 1) = \frac{3}{2}$ and $z_P(-1, 0) = \frac{1}{2}$). $\square$

We now turn to the existence and construction of reduced integral forms.

Theorem 2 *Let Q be an integral polyhedron and let P be any rational polyhedron such that $m(P) = m(Q)$ and $\text{conv}(P \cap \mathbb{Z}^E) = Q$. If the inequality $ax \leq a_0$ induces a facet F of Q , then there exists an inequality $a'x \leq a'_0$ in reduced integral form which induces F . Moreover, if it is possible to optimize over P in polynomial time, then $a'x \leq a'_0$ can be computed in polynomial time.*

Proof. Without loss of generality we may assume that $(a, a_0) \in \mathbb{Z}^E \times \mathbb{Z}$. If condition (iii) is not satisfied, then let n be the smallest positive integer for which there exists $z \in \mathbb{Z}^E$ with $az = n$ and $Cz = 0$, where $Cx = d$ is any linear system representing the equality system for Q . Since $ax \leq a_0$ is facet-inducing, a is linearly independent of the rows of C , so that matrix

$$A = \begin{bmatrix} a \\ C \end{bmatrix}$$

has full row rank. Theorem 5.2 of Schrijver [26] implies that there exists $z \in \mathbb{Z}^E$ with $az = n$ and $Cz = 0$ if and only if $nH^{-1}e$ is integral, where $[H|0]$ is the Hermite normal form of A and e is the first unit vector. By choice of n , the components of $nH^{-1}e$ are relatively prime integers, so there exists $w \in \mathbb{Z}^{m(Q)+1}$ such that $w(nH^{-1}e) = 1$. This implies that $wH^{-1} = (\frac{1}{n}, \lambda)$ for some $\lambda \in \mathbb{R}^{m(Q)}$. Theorem 5.2 of [26] also implies that $A = [H|0]U$ for some unimodular (and hence integral) matrix U , and thus

$$a' \doteq \frac{1}{n}a + \lambda C = wH^{-1}A = wH^{-1}[H|0]U = wU$$

is integral. If $z \in \mathbb{Z}^E$ has $az = n$ and $Cz = 0$, then $a'z = \frac{1}{n}az + \lambda Cz = 1$. Hence $a'x \leq a'_0 = \frac{1}{n}a_0 + \lambda d$ is a reduced integral form for $ax \leq a_0$.

Edmonds et al. [13] show how to compute a matrix C and vector d for which $Cx = d$ is the equality system for P in polynomial time, provided it is possible to optimize over P in polynomial time. Domich et al. [12] show how to compute the Hermite normal form $[H|0]$ of A in polynomial time. Hence $H^{-1}e$ can be determined in polynomial time, by solving $Hx = e$. Theorem 5.1a of [26] implies that $w \in \mathbb{Z}^{m(Q)+1}$ such that $w(nH^{-1}e) = 1$ can be determined in polynomial time using the Euclidean algorithm. Since $\lambda \in \mathbb{R}^{m(Q)}$ can be determined in polynomial time by solving $yH = w$, a reduced integral form $a'x \leq a'_0$ can be computed in polynomial time. $\square$

The following example illustrates Theorem 2 and its proof.

Example 5: With polyhedron P of Example 3, let $Q = P_I$ and $F = \{(4, 1)\}$. Here $a = (1, 0)$ and so the matrix

$$A = \begin{bmatrix} a \\ C \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix}.$$

The Hermite normal form of A is

$$H = \begin{bmatrix} 1 & 0 \\ -1 & 3 \end{bmatrix} \implies H^{-1} = \begin{bmatrix} 1 & 0 \\ \frac{1}{3} & \frac{1}{3} \end{bmatrix}.$$

Here $n = 3$ and hence w_1 and w_2 must solve $3w_1 + w_2 = 1$. If we let $w = (0, 1)$ then $\lambda = \frac{1}{3}$ and $a' = \frac{1}{3}(1, 0) + \frac{1}{3}(2, 3) = (11)$. Thus $x_1 + x_2 \leq \frac{1}{3}(4) + \frac{1}{3}(11) = 5$ is an inequality in reduced integral form which induces F . Other choices of w lead to the reduced integral forms $5x_1 + 7x_2 \leq 27$ and $-x_1 - 2x_2 \leq -6$. $\square$

Combining Theorems 1 and 2, we obtain:

Corollary 3 *Let Q be any integral polyhedron and let P be any rational polyhedron such that $\text{conv}(P \cap \mathbb{Z}^E) = Q$.*

(A) *If $m(P) = m(Q)$ then for every facet F of Q , there exists an inequality $ax \leq a_0$, inducing facet F and with integral components, such that equivalence (2) holds.*

(B) *If Q is full dimensional then every facet-inducing inequality $ax \leq a_0$ for Q , scaled so its components are relatively prime integers, verifies equivalence (2).*

Proof. For (B), recall that, since Q is a rational polyhedron, every facet-inducing inequality for Q may be scaled so its components are relatively prime integers. Since $m(Q) = 0$, condition (iii) then holds trivially so the inequality is in reduced integral form. The rest of the proof follows directly from Theorems 1 and 2. $\square$

Thus, to prove that the Chvátal rank $r_P(a, a_0)$ of a facet-inducing inequality $ax \leq a_0$ for a full-dimensional polyhedron Q is at least two, it suffices to scale the inequality so the components of a are relative prime integers, and then to find a point $\bar{x} \in P$ such that $a\bar{x} \geq a_0 + 1$, where P is any rational polyhedron

such that $\text{conv}(P \cap \mathbb{Z}^E) = Q$. In Giles and Trotter [18], the stable-set polytope is full dimensional and the facet-inducing inequality in question has coefficients of 1, 2 and 3, so equivalence (2) holds by Corollary 3(B).

We now consider the case where Q is not full-dimensional. This case is important for many combinatorial optimization problems, such as the travelling salesman problems discussed in the next sections. Although the proof of Theorem 2 is constructive, it may not be easy to apply directly. We present below a simple condition for obtaining a reduced integral form when Q is not full-dimensional.

Consider any $B \subseteq E$ such that $|B| = m(Q)$ and the $|B| \times |B|$ submatrix C_B of C with columns indexed by B is nonsingular. Subset B (and by extension, matrix C_B) is called a *basis* of the equality system. Defining $N \doteq E \setminus B$, we may write $C = (C_B, C_N)$ and, for any vector $a \in \mathbb{R}^E$, $a = (a_B, a_N)$. An inequality $ax \leq a_0$ is in *B -canonical form* if $a_B = 0$. It is in *integral B -canonical form* if the components of a_N are relatively prime integers. Note that, for any given basis B of the equality system for a rational polyhedron Q , every facet of Q is induced by a *unique* inequality in integral B -canonical form. (Indeed, the equations $a_B = 0$ define the facet-inducing inequality $ax \leq a_0$ uniquely, up to a positive multiple; this multiple is determined by the requirement that the components of a be relatively prime integers.)

Define a basis B of $Cx = d$ to be *dual-slack integral* if the matrix $C_B^{-1}C_N$ is integral. Note that this definition is equivalent to requiring that the vector $\bar{c} \doteq c - c_B C_B^{-1}C$ be integral for every integral $c \in \mathbb{Z}^E$. The vector $\bar{c}$ may thus be interpreted as that of basic dual slack variables (or reduced costs) for the linear programming problem $\max\{cx : Cx = d, x \geq 0\}$. Note also that the property of being dual-slack integral is independent of scaling and, more generally, of the linear system used to represent the given equality system. (Indeed, if $C'x = d'$ represents the same system as $Cx = d$, then $(C', d') = M(C, d)$ where M is a nonsingular matrix; it follows that $c - c_B(C'_B)^{-1}C' = \bar{c}$.)

Proposition 4 *Let B be a dual-slack integral basis of the equality system of an integral polyhedron Q . Then the integral B -canonical form of any facet-inducing inequality is a reduced integral form.*

Proof. Let B be a dual-slack integral basis of the equality system for integral polyhedron Q . Let $ax \leq a_0$ be a facet-inducing inequality in integral B -canonical form. We only need to verify condition (iii). Since the components of a_N are relatively prime, there exists $z_N \in \mathbb{Z}^N$ such that $a_N z_N = 1$. Define $z_B = -C_B^{-1}C_N z_N$ so that $Cz = C_B z_B + C_N z_N = 0$. Then z is the “certificate” in condition (iii). $\square$

Remark. This result can also be interpreted in terms of the Hermite normal form of the matrix $A = (A_B, A_N)$. If $ax \leq a_0$ is in B -canonical form for a dual-slack integral basis B , then $C_B^{-1}C_N$ is integral and $a_B = 0$ so integer multiples of the columns of A_B can be added to the columns of A_N to zero out C_N . But

then since the components of a_N are relatively prime, the Hermite normal form of A will be $[H|0]$, where

$$H = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & H_B \end{array} \right]$$

and H_B is the Hermite normal form of C_B .

Thus if P is a rational polyhedron with the same equality set as an integral polyhedron Q with $\text{conv}(P \cap \mathbb{Z}\mathbb{Z}^E) = Q$, to prove that the Chvátal rank $r_P(a, a_0)$ of a facet-inducing inequality $ax \leq a_0$ with relatively prime integer components is at least two, it suffices to exhibit a dual-slack integral basis comprised only of edges e with $a_e = 0$, and a point $x^* \in P$ with $ax^* \geq a_0 + 1$. In Balas and Saltzman [2], the (axial) three-index assignment polytope is not full dimensional, but the facet-inducing inequality induced by the clique $\{(3, 3, 3), (2, 2, 3), (2, 3, 2), (3, 2, 2)\}$ of $K_{n,n,n}$ is in B -canonical form with respect to the dual-slack integral basis described in the proof of Lemma 3.1 *ibid*. Hence equivalence (2) holds for this class of inequalities.

Further examples of integral B -canonical forms for which Proposition 4 applies will be presented in the following sections, for integral polytopes related to symmetric and asymmetric travelling salesman problems, respectively. Note, however, that dual-slack integral bases need not exist (see, e.g., Example 3), so Proposition 4 will not always apply.

3 Inequalities for Symmetric Travelling Salesman Polytopes

In this section, we consider the symmetric travelling salesman (STS) problem on the n -node complete graph $K_n = (V, E)$. The *subtour polytope* is

$$P \doteq \{x \in \mathbb{R}^E : Cx = d, x \geq 0 \text{ and } x(\delta(S)) \geq 2 \text{ for all } S \subset V \text{ with } 2 \leq |S| \leq n-1\} \quad (3)$$

where C is the node-edge incidence matrix of K_n , d is a column vector of 2's, and $\delta(S)$ denotes the set of all edges with exactly one endpoint in S . The *symmetric travelling salesman polytope* is $Q \doteq \text{conv}(P \cap \mathbb{Z}\mathbb{Z}^E)$. Since $P = Q$ for $n \leq 5$ (see, e.g., [19]), we assume $n \geq 6$.

Recall that the n degree constraints $x(\delta(v)) = 2$ are linearly independent and represent the equality system $Cx = d$ for both P and Q .

The following proposition characterizes the dual-slack integral bases of C .

Proposition 5 *A subset $B \subseteq E$ defines a dual-slack integral basis if and only if the subgraph $G_B \doteq (V, B)$ of K_n is connected, contains exactly one cycle, and this cycle is odd.*

Proof. Recall that, since graph K_n is connected and contains some odd cycles, B is a basis of its incidence matrix C if and only if every connected component of G_B contains exactly one cycle and every such cycle is odd (see, e.g., [22]).

Furthermore, let $D(v)$ denote the connected component of G_B containing node v , and let $u^v \in \mathbb{R}^V$ denote the v -th unit vector (where $u_i^v = 1$ if $i = v$, and zero otherwise). Then the unique solution y^v to $C_B y^v = u^v$ is obtained as follows: y_e^v alternates between values $+1$ and -1 for e on the path from node v to the odd cycle in $D(v)$; y_e^v alternates between values in $\{\frac{1}{2}, -\frac{1}{2}\}$ for e on this odd cycle; and $y_e^v = 0$ for all other edges. Consider a column $c^{ij} = u^i + u^j$ in C_N , so the unique solution to $C_B y = c^{ij}$ is $y^{ij} \doteq y^i + y^j$. If $D(i) \neq D(j)$, then y^{ij} has a fractional component for every edge in the odd cycles of $D(i)$ and $D(j)$. Otherwise, $D(i) = D(j)$ and, since $y_e^{ij} \equiv \frac{1}{2} + \frac{1}{2} \equiv 0$ modulo 1 for every edge e in the odd cycle of $D(i)$, all $y_e^{ij} \in \{0, \pm 1, \pm 2\}$. The proposition follows. $\square$

This proposition allows us to apply Proposition 4 and Theorem 1 so as to prove that certain classes of facet-inducing inequalities for the symmetric travelling salesman polytope have Chvátal rank at least two, relative to the subtour polytope. This approach was used in [5] to show that *ladder* inequalities have Chvátal rank at least (in fact, exactly) two, relative to the subtour polytope. We now give two additional examples validating Chvátal rank statements mentioned in the Introduction.

First, consider the class of *clique tree* inequalities. A *clique tree* is collection of two types of subsets of V , *teeth* and *handles*, satisfying certain properties; see, e.g., [6], [19], [20] or [23] for details and a definition of the corresponding STS inequality. Clique trees with no handle have a single tooth, and the corresponding STS inequalities are equivalent to constraints used in the definition of the subtour polytope P . Clique trees with one handle are called *combs*. Clique tree inequalities were introduced and proved to be facet-inducing for the STS polytope by Grötschel and Pulleyblank [20]. They were further analyzed by Boyd and Pulleyblank [6]. Chvátal et al. [11] construct a class of clique trees with any positive number h of handles, for which they prove a lower bound $f(h)$ on the Chvátal rank, growing almost linearly with h . It takes, however, at least 29 handles (and therefore at least 146 nodes) to get $f(h) > 1$, that is, to prove by their approach that the Chvátal rank is at least two, relative to the subtour polytope. We use Proposition 5 and the results from Sect. 2 above in the proof of the following result, which was first announced without proof in [20] (p. 568), and then stated as Theorem 3.3 and partly proven in [6].

Theorem 6 *A clique tree inequality has Chvátal rank one, relative to the STS subtour polytope, if and only if it is a comb.*

Proof. Theorem 3.2 in [6] states that a clique tree has Chvátal rank zero if and only if it consists of a single tooth. Theorem 2.2 *ibid.* implies that the Chvátal rank is one if the clique tree is a comb. So, we only need to prove that every clique tree with at least two handles has Chvátal rank at least two. Let H denote the union of all the handles, and let T_1 and T_2 be any two distinct teeth. There exist two nodes $t_1 \in T_1 \setminus H$ and $t_2 \in T_2 \setminus H$, and a node $h \in V \setminus (T_1 \cup T_2)$. Define

$$B = \{t_1 v : v \in V \setminus T_1\} \cup \{t_2 v : v \in T_1\} \cup \{t_2 h\} .$$

Thus, B forms a dual-slack integral basis, with odd cycle (t_1, t_2, h) . The clique tree inequality $ax \leq a_0$, as defined in equation (2.1) *ibid.*, is in integer B -canonical form. Hence, by Proposition 4, it is a reduced integral form. By Theorem 2.2 *ibid.*, there exists $x^* \in P$ with $ax^* \geq a_0 + 1$. The result then follows by applying Proposition 4 and Theorem 1. $\square$

Now consider the case $n = 7$ and the *envelope* inequality $ax \leq 8$, as given on p. 261 and in Figure 1 of [4]. Therein, the inequality is shown to be facet-inducing (note that this result only holds for $n = 7$). A dual-slack integral basis B , for which the inequality is in integral B -canonical form, is

$$B = \{(3, 4), (3, 5), (3, 6), (3, 7), (2, 4), (1, 5), (5, 6)\}.$$

A solution x^* in P with $ax^* = 9$ is given on page 265, *ibid.* It then follows that the envelope inequality for $n = 7$ is of Chvátal rank at least two, relative to the subtour polytope, as stated in [4].

Such results may easily be extended to other classes of STS facet-inducing inequalities. This is left to the interested reader.

4 Inequalities for Asymmetric Travelling Salesman Polytopes

We now turn to the asymmetric travelling salesman (ATS) problem on the n -node complete digraph $D_n = (V, A)$. The *subtour polytope* is now

$$P \doteq \{x \in \mathbb{R}^A : Cx = d, x(\delta^+(S)) \geq 1 \text{ and } x(\delta^-(S)) \geq 1 \text{ for all } S \subset V \text{ with } 2 \leq |S| \leq n-1\} \quad (4)$$

where C is the node-arc incidence matrix of D_n , d is a column vector of 1's, and $\delta^+(S)$ (resp., $\delta^-(S)$) denotes the set of all arcs with tail (resp., head) in S and head (resp., tail) in $V \setminus S$. The *asymmetric travelling salesman polytope* is $Q \doteq \text{conv}(P \cap \mathbb{Z}^A)$. We assume $n \geq 5$ (see [19] for the cases $n \leq 4$). Recall (*ibid.*) that any $2n-1$ of the $2n$ degree constraints $x(\delta^+(v)) = x(\delta^-(v)) = 1$ are linearly independent and represent the equality system for both P and Q . When $x \in \mathbb{R}^A$ satisfies these equations, the constraints $x(\delta^+(S)) \geq 1$ and $x(\delta^-(V \setminus S)) \geq 1$ are equivalent.

Recall also that the constraint matrix C is totally unimodular. Any basis B of C is therefore dual-slack integral. Bases of C may be described as follows. Construct a complete bipartite graph $K_{n,n} \doteq (V^1 \cup V^2, \tilde{A})$ where V^1 and V^2 are two copies of V , and $\tilde{A}$ consists of all edges $i^1 j^2$ with $i^1 \in V^1$ and $j^2 \in V^2$. To any subset $B \subseteq A$, associate the subset $\tilde{B} \subseteq \tilde{A}$ where $ij \in B$ if and only if $i^1 j^2 \in \tilde{B}$. Then B is a basis of C if and only if $\tilde{B}$ defines a spanning tree of $K_{n,n}$ (see, e.g., [22]).

The next proposition relates dual-slack integral bases for symmetric travelling salesman (STS) polytopes to those for the ATS polytope on the same node set V .

Proposition 7 *Let $B' = T' \cup \{i_0j_0\}$ be a basis of the equality system for the symmetric travelling salesman polytope on n nodes, where T' is a spanning tree of K_n . Let $T = \{ij, ji : ij \in T'\} \subset \tilde{A}$. Then $B = T \cup \{i_0j_0\}$ is a (dual-slack integral) basis of the equality set of the ATS polytope.*

Proof. The tree T' in K_n is a bipartite graph, and induces a partition $(S, \bar{S})$ of the node set V , where $ij \in T'$ implies that i and j are not in the same subset S or $\bar{S}$. The arc set $\tilde{T} \subset \tilde{A}$ thus induces a forest in $K_{n,n}$ comprised of exactly two connected components: one spanning the node subset $S^1 \cup \bar{S}^2$ and the other one spanning its complement $S^2 \cup \bar{S}^1$. Since the edge i_0j_0 creates an odd cycle with T' , the arc $i_0^1j_0^2$ connects these two subtrees into a spanning tree $(V^1 \cup V^2, \tilde{B})$ in $K_{n,n}$. The result follows. $\square$

We may first apply Proposition 7, in the spirit of the work in [24], to symmetric ATS inequalities. A *symmetric* ATS inequality $ax \leq a_0$ is one that satisfies $a_{ij} = a_{ji}$ for all $ij \in A$. The *corresponding STS inequality* $a'y \leq a_0$ is defined by $a'_{ij} = a_{ij}$ for all $i, j \in V$ (with $i < j$). Recall that a symmetric inequality is valid for ATS polytope if and only if the corresponding STS inequality is valid for the STS polytope.

Corollary 8 *Let $ax \leq a_0$ be a symmetric, facet-inducing ATS inequality with relative prime components, and such that the corresponding STS inequality $a'y \leq a_0$ is in integral B' -canonical form relative to a dual-slack integral basis B' for the STS polytope, and there exists y^* in the STS subtour polytope such that $a'y^* \geq a_0 + 1$. Then $ax \leq a_0$ is of Chvátal rank at least two, relative to the ATS subtour polytope.*

Proof. By Proposition 7, we may construct a (dual-slack integral) basis B for Q such that $ax \leq a_0$ is in integral B -canonical form. The point $x^* \in \mathbb{R}^A$ defined by $x_{ij}^* = x_{ji}^* = \frac{1}{2}y_{ij}^* \forall ij \in E$, satisfies $x^* \in P$ and $ax^* \geq a_0 + 1$. The result then follows by applying Proposition 4 and Theorem 1. $\square$

Note that the assumption that the symmetric inequality $ax \leq a_0$ be ATS facet-inducing implies that the corresponding STS inequality $a'y \leq a_0$ is also facet-inducing [17] for the STS polytope. The other assumptions in Corollary 8 then imply that $a'y \leq a_0$ also has Chvátal rank at least two, relative to the STS subtour polytope.

Using Theorem 6 above and Fischetti's result [17] that all clique trees are ATS facet-inducing for $n \geq 7$, Corollary 8 implies:

Corollary 9 *A clique tree inequality has Chvátal rank one, relative to the ATS subtour polytope, if and only if it is a comb.*

Remark. Corollary 9 actually implies the assertion in Theorem 6 that every clique tree inequality with at least two handles has Chvátal rank at least two with respect to the STS subtour polytope. In fact, if $ax \leq a_0$ is a symmetric ATS inequality, then the Chvátal rank of the corresponding STS inequality with

respect to the STS subtour polytope is an upper bound on the Chvátal rank of $ax \leq a_0$ with respect to the ATS subtour polytope (see Lemma 2.2.1 of Hartmann [21]).

Proposition 7 may also apply to asymmetric ATS inequalities. We conclude this paper by investigating an asymmetric lifting method due to Fischetti [16]. Let $ax \leq a_0$ be a valid inequality for Q and assume that:

- (a) its components are nonnegative integers;
- (b) there exist two distinct *isolated* nodes p and q , i.e., such that $a_{hi} = a_{ih} = 0$ for all $i \in V$ and $h \in \{p, q\}$; and
- (c) there exists a node $w \in V \setminus \{p, q\}$ such that

$$\max \{a_{iw} + a_{wj} - a_{ij} : i, j \in V \setminus \{w\}\} = 1$$

and the restriction

$$a^w x \doteq \sum (a_{ij} x_{ij} : ij \in A \setminus \delta(w)) \leq a_0 - 1$$

of $ax \leq a_0$ to $V \setminus \{w\}$ is valid and induces a nonempty face of the ATS polytope on the node set $V \setminus \{w\}$.

Applying T -lifting to $ax \leq a_0$ yields the following T -lifted inequality:

$$\beta x \doteq ax + x_{pw} + x_{wq} + x_{pq} \leq \beta_0 \doteq a_0 + 1 . \quad (5)$$

We now give a sufficient condition for the T -lifted inequality to have Chvátal rank at least two, as was claimed for an example in [16]. We shall use the following notation in the rest of this paper. Let $\tilde{V} \doteq V \setminus \{p, q\}$. We use tildes to denote any object associated with $\tilde{V}$. Thus $\tilde{A}$ is the associated arc set; $\tilde{P}$ and $\tilde{Q}$ are the subtour polytope and ATS polytope, respectively, defined on $\tilde{V}$; $\tilde{a}\tilde{x} \leq a_0$ is the restriction to $\mathbb{R}^{\tilde{A}}$ of $ax \leq a_0$; and $\tilde{\delta}(w) \doteq \tilde{\delta}^+(w) \cup \tilde{\delta}^-(w)$ is the set of all arcs in $\tilde{A}$ incident with w .

Theorem 10 *Assume that the T -lifted inequality $\beta x \leq \beta_0$ defined in (5) above, is facet-inducing for the ATS polytope Q . Assume also that there exist $\tilde{x} \in \tilde{P}$ and $\xi \in \mathbb{R}^{\tilde{\delta}(w)}$ such that $0 \leq \xi \leq \tilde{x}$,*

$$\sum_{i \neq w} \xi_{wi} = \sum_{i \neq w} \xi_{iw} = \frac{1}{2} \quad \text{and} \quad \sum_{i \neq w} \tilde{a}_{wi}(\tilde{x}_{wi} - \xi_{wi}) + \sum_{i \neq w} \tilde{a}_{iw}(\tilde{x}_{iw} - \xi_{iw}) \leq \tilde{a}\tilde{x} - a_0 - \frac{1}{2} .$$

Then $\beta x \leq \beta_0$ has Chvátal rank at least two.

Proof. Since $\beta x \leq \beta_0$ is facet-inducing with relatively prime integer components, we only need to exhibit a dual-slack integral basis $\underline{B}$ such that $\beta_e = 0 \ \forall e \in B$, and a point $x^* \in P$ with $\beta x^* \geq a_0 + 2$. Choose $r \in \tilde{V} \setminus \{w\}$ and define

$$B_w \doteq \left\{ pv, vp : v \in \tilde{V} \setminus \{w\} \right\} \cup \{qr, rq, qp\} .$$

Then B_w satisfies the conditions of Proposition 7 relative to the node set $V \setminus \{w\}$, and therefore defines a basis of the corresponding STS polytope. Adding the two arcs $wp \in \delta^+(w)$ and $qw \in \delta^-(w)$ thus yields a (dual-slack integral) basis $B \doteq B_w \cup \{wp, qw\}$ for Q , for which $\beta x \leq \beta_0$ is in integral B -canonical form. Define $x^* \in \mathbb{R}^A$ as follows:

$$x_e^* \doteq \begin{cases} \tilde{x}_e & \text{for } e \in \tilde{A} \setminus \tilde{\delta}(w) \\ \tilde{x}_e - \xi_e & \text{for } e \in \tilde{\delta}(w); \\ \xi_{iw} & \text{for } e = ip \text{ with } i \in \tilde{V} \setminus \{w\}; \\ \xi_{wi} & \text{for } e = qi \text{ with } i \in \tilde{V} \setminus \{w\}; \\ \frac{1}{2} & \text{for } e \in \{pw, wq, pq\}; \text{ and} \\ 0 & \text{for } e \in \{wp, qw, qp\}. \end{cases}$$

Since x^* satisfies the degree constraints $Cx^* = d$ and

$$\beta x^* = \tilde{a}\tilde{x} + \sum_{i \neq w} \tilde{a}_{wi}(\tilde{x}_{wi} - \xi_{wi}) + \sum_{i \neq w} \tilde{a}_{iw}(\tilde{x}_{iw} - \xi_{iw}) + \frac{3}{2} \geq a_0 + 2, \quad$$

we only need to show that x^* satisfies, for all nonempty $S \subset V$, the subtour elimination constraint $x^*(\delta^+(S)) \geq 1$. First, it is straightforward to verify that these constraints are satisfied for all nonempty $S \subset \{w, p, q\}$ and $S \supseteq \tilde{V} \setminus \{w\}$. Next, for disjoint node sets S and T , let $x(S : T) \doteq \sum \{x_{ij} : i \in S, j \in T\}$ for all $x \in \mathbb{R}^A$. For any $\emptyset \neq T \subset \tilde{V} \setminus \{w\}$, let $\bar{T} \doteq \tilde{V} \setminus (T \cup \{w\})$, so $V = T \cup \bar{T} \cup \{w, p, q\}$ and these three subsets are disjoint. We have

$$\begin{aligned} 1 &\leq \tilde{x}(T : \bar{T} \cup \{w\}) = x^*(T : \bar{T} \cup \{w, p, q\}) \\ &< x^*(T \cup \{q\} : \bar{T} \cup \{w, p\}) \leq x^*(T \cup \{p, q\} : \bar{T} \cup \{w\}) \end{aligned}$$

where the strict inequality follows from $x^*(T : q) = 0$ and $x^*(q : p) = \frac{1}{2}$, and the last inequality from $x^*(T : p) \leq \frac{1}{2} = x^*(p : w)$. Hence the subtour elimination constraints $x^*(\delta^+(S)) \geq 1$ are satisfied for $S = T$; $S = T \cup \{q\}$; and $S = T \cup \{p, q\}$. Since $x^*(T : p) < 1 = x^*(p : w) + x^*(p : q)$, we have $x^*(\delta^+(T \cup \{p\})) \geq 1$. By exchanging $\bar{T}$ and T , and also p and q , it follows similarly that $x^*(\delta^+(S)) = x^*(\delta^-(\bar{S})) \geq 1$ holds for $S = T \cup \{w\}$; $S = T \cup \{w, p\}$; $S = T \cup \{w, q, p\}$; and $S = T \cup \{w, q\}$. The proof is complete. $\square$

We now apply T -lifting to any clique tree inequality. In a clique tree, a tooth is *pendent* if it intersects exactly one handle, and *nonpendent* otherwise, i.e., if it intersects at least two handles. Let H denote the union of all handles. The following proposition gives two sufficient conditions for a T -lifted clique tree inequality to have Chvátal rank at least 2, relative to the ATS subtour polytope. We refer to [16] for a discussion of cases where the T -lifted clique tree inequality is facet-inducing for the ATS polytope.

Proposition 11 *A T -lifted clique tree inequality (5) has Chvátal rank at least 2, relative to the ATS subtour polytope, if it is facet-inducing for the ATS polytope and either*

- (a) it has at least one handle, and node w is in $T \cap H$ where T is a pendent tooth; or
 (b) it has at least two nonpendent teeth.

Proof. By Theorem 10, we only need to construct the required $\tilde{x}$ and ξ . For case (a), recall that, if a clique tree has at least one handle, then there exists $\tilde{y}$ in the STS subtour polytope such that $a'\tilde{y} - a_0 \geq \frac{1}{2}$. Further, from step (6) in the proof of Theorem 2.2 in [6] (p. 170), for every w as described in (a), we can construct such a $\tilde{y}$ with $\tilde{y}_{wk} = \tilde{y}_{wl} = \frac{1}{2}$ for two edges wk and wl with $a'_{wk} = a'_{wl} = 0$. Then $\tilde{x}$ defined by $\tilde{x}_{ij} = \tilde{x}_{ji} = \tilde{y}_{ij}$ for all $ij \in E$, and ξ defined by $\xi_{ij} = \frac{1}{4}$ if $ij \in \{wk, kw, wl, lw\}$, and 0 otherwise, satisfy the conditions of Theorem 10. The proof is complete for case (a). Now consider case (b). The solution $\tilde{y}$ constructed in the proof of Theorem 2.2 in [6] satisfies $a'\tilde{y} - a_0 \geq \frac{3}{2}$. If $w \in T \setminus H$ where T is a pendent teeth which intersects exactly two handles, then $\tilde{y}_e = 1$ for two edges wk and wl with $k, l \in T$; and 0 for all other edges (see step (3) in the proof of Theorem 2.2 *ibid.*). With $\tilde{x}$ as defined above, let the only nonzero components of ξ be $\xi_{wk} = \xi_{kw} = \frac{1}{2}$. For every choice of w other than an isolated node, that in (a) or that just discussed, we can apply the construction in the proof of Theorem 2.2 in [6] so that the solution $\tilde{y}$ satisfies $\tilde{y}_{wk} = \tilde{y}_{wl} = \frac{1}{2}$ for two nodes k, l in the same handle as w , or in the same nonpendent tooth T (intersecting at least three handles) as w . With $\tilde{x}$ as defined above, let the only nonzero components of ξ be $\xi_{wk} = \xi_{kw} = \xi_{wl} = \xi_{lw} = \frac{1}{4}$. Finally, note that $a_{ij} = a'_{ij} = 1$ for all arcs $ij \in A$ such that $\xi_{ij} > 0$, so ξ satisfies all the requisite conditions. The proof is complete. $\square$

In particular, case (a) of Proposition 11 proves a statement in [16] (p. 263) that a T -lifted comb, where node w is in a tooth and not in the handle, has Chvátal rank at least two, relative to the ATS polytope.

Acknowledgements. The authors would like to thank Professor William H. Cunningham, of the University of Waterloo, for his support and his encouragement to complete this work, and Professor Sebastian Ceria, of Columbia University, for information regarding the disjunctive rank of inequalities.

References

1. E. Balas, S. Ceria and G. Cornuéjols (1993), "A lift-and-project cutting plane algorithm for mixed 0-1 programs," *Mathematical Programming* **58**, 295–324.
2. E. Balas and M.J. Saltzman (1989), "Facets of the three-index assignment polytope," *Discrete Applied Mathematics* **23**, 201–229.
3. A. Bockmayr, F. Eisenbrand, M. Hartmann and A.S. Schulz (to appear), "On the Chvátal rank of polytopes in the 0/1 cube," to appear in *Discrete Applied Mathematics*.
4. S.C. Boyd and W.H. Cunningham (1991), "Small travelling salesman polytopes," *Mathematics of Operations Research* **16**, 259–271.
5. S.C. Boyd, W.H. Cunningham, M. Queyranne and Y. Wang (1995), "Ladders for travelling salesmen," *SIAM Journal on Optimization* **5**, 408–420.

6. S.C. Boyd and W.R. Pulleyblank (1991), "Optimizing over the subtour polytope of the travelling salesman problem," *Mathematical Programming* **49**, 163–187.
7. A. Caprara and M. Fischetti (1996), "0-1/2 Chvátal-Gomory cuts," *Mathematical Programming* **74**, 221–236.
8. A. Caprara, M. Fischetti and A.N. Letchford (1997), "On the separation of maximally violated mod- k cuts," DEIS-OR Technical Report OR-97-8, Dipartimento di Elettronica, Informatica e Sistemistica Università degli Studi di Bologna, Italy, May 1997.
9. S. Ceria (1993), "Lift-and-project methods for mixed 0-1 programs," Ph.D. dissertation, Carnegie-Mellon University.
10. V. Chvátal (1973), "Edmonds polytopes and a hierarchy of combinatorial problems," *Discrete Mathematics* **4**, 305–337.
11. V. Chvátal, W. Cook and M. Hartmann (1989), "On cutting-plane proofs in combinatorial optimization," *Linear Algebra and Its Applications* **114/115**, 455–499.
12. P.D. Domich, R. Kannan and L.E. Trotter, Jr. (1987), "Hermite normal form computation using modulo determinant arithmetic," *Mathematics of Operations Research* **12**, 50–59.
13. J. Edmonds, L. Lovász and W.R. Pulleyblank (1982), "Brick decompositions and the matching rank of graphs," *Combinatorica* **2**, 247–274.
14. F. Eisenbrand (1998), "A note on the membership problem for the elementary closure of a polyhedron," Preprint TU-Berlin No. 605/1998, Fachbereich Mathematik, Technische Universität Berlin, Germany, November 1998.
15. F. Eisenbrand and A.S. Schulz (1999), "Bounds on the Chvátal Rank of Polytopes in the 0/1-Cube", this Volume.
16. M. Fischetti (1992), "Three facet-lifting theorems for the asymmetric traveling salesman polytope", pp. 260–273 in E. Balas, G. Cornuéjols and R. Kannan, eds., *Integer Programming and Combinatorial Optimization* (Proceedings of the IPCO2 Conference), G.S.I.A., Carnegie Mellon University.
17. M. Fischetti (1995), "Clique tree inequalities define facets of the asymmetric travelling salesman polytope," *Discrete Applied Mathematics* **56**, 9–18.
18. R. Giles and L.E. Trotter, Jr. (1981), "On stable set polyhedra for $K_{1,3}$ -free graphs," *Journal of Combinatorial Theory Ser. B* **31**, 313–326.
19. M. Grötschel and M.W. Padberg (1985), "Polyhedral theory," pp. 251–305 in: E.L. Lawler et al., eds., *The Travelling Salesman Problem*, Wiley, New York.
20. M. Grötschel and W.R. Pulleyblank (1986), "Clique tree inequalities and the symmetric travelling salesman problem," *Mathematics of Operations Research* **11**, 537–569.
21. M. Hartmann (1988), "Cutting planes and the complexity of the integer hull," Ph.D. thesis, Cornell University.
22. E.L. Johnson (1965), "Programming in networks and graphs", Research Report ORC 65-1, Operations Research Center, University of California–Berkeley.
23. G.L. Nemhauser and L.A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley, New York.
24. M. Queyranne and Y. Wang (1995), "Symmetric inequalities and their composition for asymmetric travelling salesman polytopes," *Mathematics of Operations Research* **20**, 838–863.
25. A. Schrijver (1980), "On cutting planes," in: M. Deza and I.G. Rosenberg, eds., *Combinatorics 79 Part II, Ann. Discrete Math.* **9**, 291–296.
26. A. Schrijver (1986), *Theory of Linear and Integer Programming*, Wiley, Chichester.

The Square-Free 2-Factor Problem in Bipartite Graphs

David Hartvigsen

College of Business Administration
University of Notre Dame
P.O. Box 399
Notre Dame, IN 46556-0399
Hartvigsen.1@nd.edu

Abstract. The 2-factor problem is to find, in an undirected graph, a spanning subgraph whose components are all simple cycles; hence it is a relaxation of the problem of finding a Hamilton tour in a graph. In this paper we study, in bipartite graphs, a problem of intermediate difficulty: the problem of finding a 2-factor that contains no 4-cycles. We introduce a polynomial time algorithm for this problem; we also present an “augmenting path” theorem, a polyhedral characterization, and a “Tutte-type” characterization of the bipartite graphs that contain such 2-factors.

1 Introduction

A *2-factor* in a simple undirected graph is a spanning subgraph whose components are all simple cycles (i.e., the degrees of all nodes in the subgraph are 2). A *k-restricted 2-factor*, for $k \geq 2$ and integer, is a 2-factor with no cycles of length k or less. Let us denote by P_k the problem of finding a k -restricted 2-factor in a graph. Thus P_2 is the problem of finding a 2-factor in a graph and P_k , for $n/2 \leq k \leq n-1$ (where n is the number of nodes in the graph), is the problem of finding a Hamilton cycle in a graph.

The basic problem P_2 has been extensively studied. A polynomial algorithm exists due to a reduction (see Tutte [11]) to the classical matching problem (see Edmonds [6]). Also well-known is a “good” characterization of those graphs that have a 2-factor (due to Tutte [10]) and a polyhedral characterization of the convex hull of incidence vectors of 2-factors (which can also be obtained using Tutte’s reduction and Edmonds’ polyhedral result for matchings [5]). For a more complete history of this problem see Schrijver [8].

Of course a polynomial time algorithm for P_k , for $n/2 \leq k \leq n-1$, is unlikely since the problem of deciding if a graph has a Hamilton tour is NP-complete. In fact, Papadimitiou (see [2]) showed that deciding if a graph has a k -restricted 2-factor for $k \geq 5$ is NP-complete. A polynomial time algorithm for P_3 was given in [7]. This leaves P_4 whose status is open.

In this note we consider the problem P_4 on bipartite graphs, which we call the *bipartite square-free 2-factor problem*. (The idea of studying this problem

was suggested to the author by Cunningham and Geelen [3].) Observe that for bipartite graphs the problems P_k of interest are for $k \geq 2$ and even. We introduce here a polynomial time algorithm for the bipartite square-free 2-factor problem. The algorithm uses techniques similar to those used for the problem P_3 on general graphs, but the resulting algorithm is simpler. We also present in this paper, for the bipartite square-free 2-factor problem, three theorems that are the appropriate versions of the following classical results for matchings in graphs:

- Tutte’s [9] characterization of the graphs that contain perfect matchings;
- Berge’s [1] “augmenting path” characterization of maximum matchings in graphs;
- Edmonds’ [5] polyhedral characterization of matchings in graphs.

Our three theorems follow from the validity of our algorithm. We present our Tutte-type theorem in Section 2. We introduce a slightly more general version of the k -restricted 2-factor problem in Section 3, in terms of which we state our remaining results. In particular, we present our “augmenting path” theorem in Section 4 and our polyhedral theorem in Section 5. We introduce some of the ideas of the algorithm in Section 6. We contrast all of our results with the comparable results for the problem P_2 on bipartite graphs.

Before ending this section let us briefly mention two related classes of problems. First, consider the problems P_k , for $k \geq 6$, on bipartite graphs. The status of these problems is open. Second, consider the “weighted” versions of the problems P_k . That is, given a graph with nonnegative weights, let *weighted- P_k* be the problem of finding a k -restricted 2-factor such that the sum of the weights of its edges is a maximum. Vornberger [12] showed that *weighted- P_4* is NP-hard. The status of *weighted- P_3* remains open. Cunningham and Wang [4] have studied the closely related polytopes given by the convex hull of incidence vectors of k -restricted 2-factors.

2 Tutte-Type Theorems

In this section we present a theorem that characterizes those bipartite graphs that have a 2-factor and a theorem that characterizes those bipartite graphs that have a square-free 2-factor. The first theorem is essentially a bipartite version of Tutte’s 2-factor theorem (see [10]). The necessity of the condition is straightforward to establish. The sufficiency follows from a max-min theorem of Edmonds (see [5]). The sufficiency also follows from the proof of the validity of the first algorithm we present in Section 6.

For a graph $H = (V, E)$, let $A(H)$ denote the number of nodes of H that have degree 0; let $B(H)$ denote the number of nodes of H that have degree 1. For $S \subseteq V$, we let $H[S]$ denote the subgraph of H induced by S .

Theorem 1. *A bipartite graph $G = (V, E)$ has a 2-factor if and only if for every subset $V' \subseteq V$, $2|V'| \geq 2A(G[V \setminus V']) + B(G[V \setminus V'])$.*

With a few more definitions we can present a variant of this result that holds for square-free 2-factors. Again, let $H = (V, E)$ be a graph. We refer to a cycle in H of length 4 as a *square*. Let $C(H)$ denote the number of isolated squares of H (i.e., those squares all of whose nodes have degree 2 in H); and let $D(H)$ denote the number of squares of H for which two nonadjacent nodes have degree 2 (in H) but at least one of the other two nodes has degree > 2 (in H). The necessity of the condition in the following theorem is straightforward to establish. The sufficiency appears to be considerably more difficult; it follows from the validity of the second algorithm we introduce in Section 6.

Theorem 2. *A bipartite graph $G = (V, E)$ has a square-free 2-factor if and only if for every subset $V' \subseteq V$,*

$$2|V'| \geq 2A(G[V \setminus V']) + B(G[V \setminus V']) + 2C(G[V \setminus V']) + D(G[V \setminus V']).$$

3 More General Problems

We introduce in this section two problems that are slightly more general than the problems P_2 and P_4 in bipartite graphs. These are the problems we consider in the remainder of this note. In particular, these are the problems that our algorithms in Section 6 actually solve. Nevertheless, our algorithms also solve P_2 and P_4 in bipartite graphs.

Consider a simple undirected graph G . A *simple 2-matching in G* is a subgraph of G whose components are simple paths and cycles (i.e., the degrees of all nodes in the subgraph are 1 or 2). The *cardinality* of a simple 2-matching is its number of edges. A *square-free simple 2-matching* is a simple 2-matching that contains no cycles of length 4. We subsequently consider in this note the following two problems on bipartite graphs: finding a maximum cardinality simple 2-matching and finding a maximum cardinality square-free simple 2-matching. Note that, as we claimed above, an algorithm that finds a (square-free) maximum cardinality simple 2-matching also determines if a graph has a (square-free) 2-factor.

4 Augmenting Path Theorems

Our algorithms for finding maximum cardinality simple 2-matchings in a bipartite graph are similar in form to Edmonds' algorithm for finding a maximum cardinality matching in a graph (see [6]). Edmonds' algorithm builds up such a matching incrementally by identifying "augmenting paths." In this section we define types of "augmenting paths" for our two problems and state two theorems that characterize maximum cardinality simple 2-matchings for our problems in terms of these "augmenting paths." These theorems are in the spirit of Berge's theorem [1] for matchings.

Consider a graph G and a simple 2-matching M . A node in G is called *saturated* if it is incident in G with two edges of M ; if it is incident with one edge or no edge of M it is called *deficient*.

An *augmenting path* with respect to M is a simple path (no repeated nodes) with the following properties: the edges are alternately in and out of M ; the endedges are not in M ; and the endnodes are deficient. Hence if G has an augmenting path, then a simple 2-matching with cardinality one greater than $|M|$ can be obtained by interchanging the edges in and out of M along this path. The following theorem shows that we have the converse as well. This result follows from the validity of our algorithm for finding maximum cardinality simple 2-matchings in bipartite graphs. It also follows from Berge's [1] augmenting path theorem for matchings with the reduction of Tutte [11].

Theorem 3. *A simple 2-matching M in a bipartite graph G has maximum cardinality if and only if G contains no augmenting path with respect to M .*

We next consider the case of square-free simple 2-matchings. Let M denote a square-free simple 2-matching in G . An *augmenting structure* with respect to M is a simple path (no repeated nodes) P together with a collection (possibly empty) of cycles $C_1, \dots, C_p$ with the following properties:

- $P, C_1, \dots, C_p$ are pairwise edge-disjoint (they may share nodes);
- For P : the edges are alternately in and out of M ; the endedges are not in M ; and the endnodes are deficient.
- For each C_i : the edges are alternately in and out of M ;
- Interchanging the edges of $P, C_1, \dots, C_p$ in and out of M results in a square-free simple 2-matching in G .

Hence if G has an augmenting structure, then a square-free simple 2-matching with cardinality one greater than $|M|$ can be obtained by interchanging the edges in and out of M in this structure. The following theorem shows that we have the converse as well. The proof of this result follows from the validity of our algorithm for finding maximum cardinality square-free simple 2-matchings in bipartite graphs. The example following the theorem shows why the definition of augmenting paths is not sufficient for the square-free case.

Theorem 4. *A square-free simple 2-matching M in a bipartite graph G has maximum cardinality if and only if G contains no augmenting structure with respect to M .*

Example Consider the square-free simple 2-matching M in Figure 1. Bold lines are edges in M ; non-bold lines are edges not in M . Let P denote the path with the six nodes $a - f$ and let $C1$ and $C2$ denote the cycles bounding the regions with the corresponding labels. Observe that P is the only augmenting path in this graph, yet if P alone is used to increase $|M|$, then a square is created in the resulting matching. However, $|M|$ can be increased by one by using the augmenting structure $P, C1$, and $C2$.

Observe that integral solutions to (P') are incidence vectors for square-free simple 2-matchings and hence an optimal integral solution is a maximum cardinality square-free simple 2-matching. However, the constraint matrix for (P') is not totally unimodular, in general.

Theorem 6. *For any bipartite graph G , (P') has an integer optimal solution (which is a maximum cardinality square-free simple 2-matching).*

This theorem can be proved from the structure available at the end of the algorithm. In particular, we can produce an integral feasible primal solution and a half-integral feasible dual solution that satisfy complementary slackness.

6 The Algorithms

We begin this section with a complete statement of the algorithm for finding a maximum cardinality simple 2-matching in a bipartite graph. This algorithm can be obtained using Tutte's reduction [11] to the matching problem and then applying Edmonds' matching algorithm [6]. We present this algorithm in the same form that is used for the square-free algorithm so that we may discuss some of the features of the square-free algorithm later in this section.

In the course of the algorithm for finding a maximum cardinality simple 2-matching in a bipartite graph, we construct a subgraph S of G called an *alternating structure*. We next describe some properties of this structure that are maintained throughout the algorithm.

The nodes of S are either *odd* or *even* and have the following properties.

- Odd nodes are incident with two edges of M (zero, one, or two of these can be in S); the other endnodes of these edges are even nodes.
- Even nodes may be saturated or deficient.
- If an even node is saturated, then exactly one of these two edges of M must be in S and the other endnode of this edge must be odd.

S is partitioned into the union of node disjoint trees. These trees have the following properties:

- Each tree has a unique node called its *root*.
- The set of roots is the same as the set of deficient nodes.
- All roots are even nodes.

Consider an arbitrary path in such a tree that contains the tree's root as an endnode. Then this path has the following properties:

- The nodes of the path are alternately even and odd.
- The edges of the path are alternately in and not in M such that the edge incident with the root is not in M .

Algorithm :**Input:** Bipartite graph G .**Output:** A maximum cardinality simple 2-matching of G .**Step 0:** Let M be any simple 2-matching of G .**Step 1:** If M is a 2-factor, done. Otherwise, let S be the set of deficient nodes of G , where these nodes are the roots of S and are even.**Step 2:** If every edge of G that is incident with an even node is either an edge of M or has an odd node as its other endnode, then M has maximum cardinality; done. Otherwise, select an edge $j \notin M$ joining an even node v to a node w that is not an odd node.**Case 1:** $w \notin S$. Go to Step 3.**Case 2:** w is an even node. Go to Step 4.**Step 3:** [Grow the structure] $w \notin S$.

w must be incident with two edges in M : wu_1 and wu_2 (otherwise w is deficient, hence a root in S). Neither u_1 nor u_2 is odd, since then w would be even. Thus each u_i is either even or $\notin S$. Make w odd and add it and the edge vw to the tree that contains v . If $u_i \notin S$, then make it even and add it and the edge wu_i to the tree that contains v . If u_i is even, do not change its status.

Step 4: [Augment the matching] w is an even node.

Observe that v and w must be in different trees (otherwise there exists an odd cycle). Consider the path consisting of the path from v to the root of its tree, the path from w to the root of its tree, plus the edge vw . The edges along this path are alternately in and not in M with the endedges not in M and the endnodes are deficient (i.e., it is an augmenting path). Interchange edges in and out of the matching along this path. Go to Step 1.

End .

The validity of this algorithm can be proved using arguments similar to those used by Edmonds [6] to prove the validity of his matching algorithm.

We next introduce some of the key ideas of our square-free version of the above algorithm by considering an example. To begin, consider an application of the above algorithm to the graph in Figure 1. Let us begin in Step 0 with the matching M illustrated: bold lines are edges in M ; non-bold lines are edges not in M . Observe that nodes a and f are the only deficient nodes, hence the algorithm starts in Step 1 by making these two nodes into even nodes and the roots of two trees in S . The algorithm may next select the edge ab in Step 2. It would then go to Step 3 and make b an odd node, and c and g even nodes. It would add the edges ab , bc , and bg to the tree containing a . The algorithm may next select the edge cd in Step 2. As above it would then make d an odd node, and e and h even nodes. It would add the edges cd , de , and dh to the tree containing a . The algorithm may next select the edge ef in Step 2. This then calls for augmenting the matching in Step 4. However, if we were to do so, we would create a square in the matching. The way in which this problem is dealt with in the square-free version of the algorithm is as follows: When we consider edge cd , we make only h an even node. Thus we only look to grow the tree from

node h . As we continue to grow the tree, we eventually discover the cycles $C1$ and $C2$, and incorporate them into the structure. When this is accomplished, we make e an even node and add the edge de to the tree containing a . Now when we consider the edge ef , we may augment by using the augmenting structure P , $C1$, $C2$ as described in the example in Section 4. In order to handle the search for these augmenting structures, the definition of alternating structure becomes more complex. Several new types of nodes are added as well as a more complex tree structure. Also, as in the classical matching algorithm of Edmonds [6], certain subgraphs (in this case squares) may be shrunk in the course of the algorithm.

Acknowledgement I would like to thank Gerard Cornuejols for some helpful discussions of this work and for his major role in developing with me in my doctoral thesis the techniques used in this paper.

References

1. Berge, C., Two theorems in graph theory. *Proceedings of the National Academy of Sciences (U.S.A.)* 43 (1957) 842-844.
2. Cornuejols, G. and W.R. Pulleyblank. A matching problem with side conditions. *Disc. Math.* 29 (1980) 135-159.
3. Cunningham, W.H. and J. Geelen. Personal communication (1997).
4. Cunningham, W.H. and Y. Wang. Restricted 2-factor polytopes. *Working paper, Dept. of Comb. and Opt., University of Waterloo* (Feb. 1997).
5. Edmonds, J., Maximum matching and a polyhedron with 0,1 vertices. *J. Res. Nat. Bur. Standards Sect. B* 69 (1965) 73-77.
6. Edmonds, J., Paths, trees, and flowers. *Canad. J. Math.* 17 (1965) 449-467.
7. Hartvigsen, D. Extensions of Matching Theory. *Ph.D. Thesis*, Carnegie-Mellon University (1984).
8. Schrijver, A., Min-max results in combinatorial optimization. in *Mathematical Programming, the State of the Art: Bonn, 1982*, Eds.: A. Bachem, M. Grotschel, and B. Korte, Springer-Verlag, Berlin (1983) 439-500.
9. Tutte, W.T., The factorization of linear graphs. *J. London Math. Soc.* 22 (1947) 107-111.
10. Tutte, W.T., The factors of graphs. *Canad. J. Math.* 4 (1952) 314-328.
11. Tutte, W.T., A short proof of the factor theorem for finite graphs. *Canad. J. Math.* 6 (1954) 347-352.
12. Vornberger, O., Easy and hard cycle covers. *Preprint*, Universitat Paderborn (1980).

The m-Cost ATSP

Christoph Helmberg

Konrad-Zuse-Zentrum Berlin

Takustraße 7, 14195 Berlin, Germany

helmberg@zib.de, <http://www.zib.de/helmberg>

Abstract. Although the m-ATSP (or multi traveling salesman problem) is well known for its importance in scheduling and vehicle routing, it has, to the best of our knowledge, never been studied polyhedrally, i.e., it has always been transformed to the standard ATSP. This transformation is valid only if the cost of an arc from node i to node j is the same for all machines. In many practical applications this is not the case, machines produce with different speeds and require different (usually sequence dependent) setup times. We present first results of a polyhedral analysis of the m-ATSP in full generality. For this we exploit the tight relation between the subproblem for one machine and the prize collecting traveling salesman problem. We show that, for $m \geq 3$ machines, all facets of the one machine subproblem also define facets of the m-ATSP polytope. In particular the inequalities corresponding to the subtour elimination constraints in the one machine subproblems are facet defining for m-ATSP for $m \geq 2$ and can be separated in polynomial time. Furthermore, they imply the subtour elimination constraints for the ATSP-problem obtained via the standard transformation for identical machines. In addition, we identify a new class of facet defining inequalities of the one machine subproblem, that are also facet defining for m-ATSP for $m \geq 2$. To illustrate the efficacy of the approach we present numerical results for a scheduling problem with non-identical machines, arising in the production of gift wrap at Herlitz PBS AG.

1 Introduction

For Herlitz PBS AG, Berlin, we are developing a software package for the following scheduling problem. Gift wrap has to be printed on two non-identical printing machines. The gift wrap is printed in up to six colors on various kinds of materials in various widths. The colors may differ considerably from one gift wrap to the next. The machines differ in speed and capabilities, not all jobs can be processed on both machines. Setup times for the machines are quite large in comparison to printing time. They depend strongly on whether material, width, or colors have to be changed, so in particular on the sequence of the jobs. The task is to find an assignment of the jobs to the machines and an ordering on these machines such that the last job is finished as soon as possible, i.e., minimize the makespan for a scheduling problem on two (non-identical) machines with sequence dependent set-up times.

An obvious mathematical model for this problem is the asymmetric multi traveling salesman problem with separate arc costs for each salesman. Although there is considerable literature on the TSP/ATSP [10,13,14,15,3,11,7] as well as on the m-ATSP for vehicle routing (see [5] and references therein) it seems that the m-ATSP problem in full generality has never been studied from a polyhedral point of view. Existing work on the m-ATSP relies on the standard transformation to ATSP (see e.g. [16] or Section 6) which assumes that all salesmen need the same time for the same route. The general case of non-identical salesmen should also be of importance in vehicle routing where usually not all vehicles in a car park have the same capabilities or fuel consumption.

In this paper we give a precise definition of what we call the m-Cost ATSP and present a ‘canonical’ integer programming formulation. This formulation includes an exponential class of inequalities, which we call the *v0-cut inequalities*. They may be interpreted as a kind of conditional subtour elimination constraints and are, in a certain sense, equivalent to the inequalities introduced in [1], Theorem 2.5, for the Price Collecting TSP (PCTSP) (see also [8,2,9]). The PCTSP is tightly related to the one machine subproblem of the m-Cost ATSP and we will clarify this relation in Section 4. Another variant, the generalized traveling salesman problem [6], also allows for skipping certain jobs but differs in that the jobs are partitioned into sets and at least one job from each set has to be visited by the tour.

The main results of the paper are: Facets of the one machine subproblem define facets of the m-Cost ATSP polytope for $m \geq 3$; the non-negativity constraints and the v0-cut inequalities are facet defining for the m-Cost ATSP polytope for $m \geq 2$, they can be separated in polynomial time, and they imply the subtour elimination constraints if the standard transformation for identical machines from m-ATSP to ATSP is applied. In addition, we identify a new class of so called *nested conflict inequalities* that are facet defining for the polytope of the one machine subproblem and the m-Cost ATSP polytope for $m \geq 2$.

For our real-world instances the linear relaxation based on the v0-cut inequalities yields, in average, a relative gap of 0.2% with respect to the best solution known. In comparison, the relaxation using subtour elimination constraints on the transformed problem exhibits an average relative gap of 0.5%, so this approach reduces the gap by 60%.

The paper is organized as follows. Section 2 introduces the necessary notation and gives the problem definition. In Section 3 we present an integer programming formulation of the m-Cost ATSP and determine the dimension of the m-Cost ATSP polytope. Section 4 is devoted to the one machine subproblem, its relation to the PCTSP, and the separation algorithm. In Section 5 the facet defining inequalities of the subproblem are shown to be facet defining for the m-Cost ATSP polytope. In Section 6 we prove that the v0-cut inequalities imply the subtour elimination constraints for the transformed problem. Finally, in Section 7 numerical results are presented for the scheduling problem at Herlitz PBS AG.

2 Notation and Problem Formulation

Let $D = (V, A)$ be a digraph on node set V and arc set A without multiple arcs but possibly with loops. An arc is an ordered pair (v, w) of nodes $v, w \in V$, v is called tail, w head of (v, w) . Most of the time we will simply write vw instead of (v, w) . For a subset $S \subseteq V$, $\delta^-(S) = \{vw \in A : v \in V \setminus S, w \in S\}$ is the set of arcs of A that have their head in S and their tail in $V \setminus S$. Similarly, $\delta^+(S) = \{vw \in A : v \in S, w \in V \setminus S\}$ is the set of arcs of A having tail in S and head in $V \setminus S$. By definition, loops are not contained in any set $\delta^-(S)$ or $\delta^+(S)$. For a node $v \in V$ we will write $\delta^-(v)$ instead of $\delta^-(\{v\})$. For an arc set $\hat{A} \subset A$, $V(\hat{A}) = \{v \in V : vw \in \hat{A} \text{ or } wv \in \hat{A}\}$ is the set of nodes that are tail or head of an arc in $\hat{A}$. For a node set $S \subset V$, $A(S) = \{vw \in A : v \in S \text{ and } w \in S\}$ is the set of arcs having both head and tail in S . For a vector $x \in \mathbb{Q}^{|A|}$ the sum of the weights over an arc set $\hat{A} \subset A$ is denoted by $x(\hat{A}) = \sum_{a \in \hat{A}} x_a$.

A dicycle is a set of $k > 0$ arcs $C = \{v_1v_2, v_2v_3, \dots, v_kv_1\} \subseteq A$ with distinct vertices $v_1, \dots, v_k \in V$. The number of arcs k is referred to as the length of the dicycle. A tour is a dicycle of length $|V|$. A loop is a dicycle of length 1.

Now consider the problem of scheduling n jobs on m distinct machines. Let $J = \{1, \dots, n\}$ denote the set of jobs and $M = \{1, \dots, m\}$ the set of machines. In order to model all possible schedules we introduce for each machine $k \in M$ a digraph $D_k^0 = (V_k, A_k)$ with $V_k = J \cup \{0\}$ and arc set $A_k = \{(0, 0)_k\} \cup \{(i, j)_k : i, j \in V_k, i \neq j\}$ (node 0 allows to model initial and final state of each machine as well as idle costs). We will write ijk instead of $(i, j)_k$. Usually we will regard the nodes in V_k and V_i as the same objects. Sometimes, however, we will have to discern between nodes in V_k and V_i , then we will do so by adding a subscript, e.g., 0_k for $0 \in V_k$. For $S \subseteq V_k$ the subscript in the symbols $\delta_k^-(S)$ and $\delta_k^+(S)$ will indicate on which arc set we are working. We will frequently need the union of all arc sets which we denote by $\mathcal{A} = \bigcup_{k=1}^m A_k$. With each arc $ijk \in \mathcal{A}$ we associate a cost or weight c_{ijk} .

A feasible solution $A_F = C_1 \cup \dots \cup C_m$ is a set of $m + n$ arcs that is the union of m dicycles $C_k \subseteq A_k$ for $k \in M$, such that $0 \in V_k(C_k)$ for all $k \in M$ and for each $v \in J$ there is a $k \in M$ with $v \in V_k(C_k)$. The set of feasible solutions is

$$\mathcal{F}_n^m = \left\{ C_1 \cup \dots \cup C_m : \sum_{k=1}^m |C_k| = n + m, \right. \\ \left. \begin{array}{l} \forall k \in M : C_k \subseteq A_k \text{ is a dicycle with } 0 \in V_k(C_k), \\ \forall v \in J : \exists k \in M : v \in V_k(C_k) \end{array} \right\}.$$

One proves easily that each job is assigned uniquely.

Proposition 1. *Let $A_F = C_1 \cup \dots \cup C_m \in \mathcal{F}_n^m$, then for $v \in J$ there is a unique $k \in M$ with $v \in V(C_k)$.*

An $A_F \in \mathcal{F}_n^m$ may be interpreted in the following way: if $ijk \in A_F$ and $i, j \in J$ then job j has to be processed directly after job i on machine k . If $i = 0$ and $j \in J$, then j is the first job to be processed on machine k . If $i \in J$ and $j = 0$

then i is the last job to be processed on machine k . If $i = j = 0$ then machine k does not process any jobs.

The optimization problem we will deal with is to find

$$(\text{m-Cost ATSP}) \quad \min \{c(A_F) : A_F \in \mathcal{F}_n^m\}.$$

It is not difficult to show that (m-Cost ATSP) is NP-complete.

3 The m -Cost ATSP Polytope

We will approach the problem from a polyhedral point of view. To this end define, for $A \subseteq \mathcal{A}$, the incidence vector $x^A \in \{0, 1\}^{|\mathcal{A}|}$ by

$$x_{ijk}^A = \begin{cases} 1, & \text{if } ijk \in A \\ 0, & \text{otherwise.} \end{cases}$$

The m-Cost ATSP polytope is defined as the convex hull of the incidence vectors of all feasible solutions, i.e.,

$$P_n^m = \text{conv} \{x^A : A \in \mathcal{F}_n^m\}.$$

In order to study this polytope consider the following set of constraints

$$x_{00k} + x(\delta_k^+(0)) = 1 \quad k \in M \quad (1)$$

$$\sum_{k \in M} x(\delta_k^+(j)) = 1 \quad j \in J \quad (2)$$

$$x(\delta_k^-(j)) = x(\delta_k^+(j)) \quad j \in J, k \in M \quad (3)$$

$$x(\delta_k^+(S)) \geq x(\delta_k^+(v)) \quad k \in M, S \subseteq J, |S| \geq 2, v \in S \quad (4)$$

$$x_{ijk} \in \{0, 1\} \quad ijk \in \mathcal{A}. \quad (5)$$

The constraints of (1) ensure that for each node 0_k , $k \in M$, one arc is selected. In (2) we require that for each job $j \in J$ one outgoing arc of j is selected among all such arcs on all machines. Constraints (3) are flow conservation constraints. Constraints (4) are a kind of conditional subtour elimination constraints that ensure that any flow through v_k in graph k reaches node 0_k in this graph (note, that $0 \notin J$), i.e., the flow through v_k has to cross the cut from S to $V_k \setminus S$. Therefore, we call constraints (4) *v0-cut inequalities*.

Lemma 1. *An incidence vector x^A of $A \subseteq \mathcal{A}$ is a feasible solution of (1)–(5) if and only if $A \in \mathcal{F}_n^m$.*

Proof. The *if* part is easy to check, so consider the *only if* part. Let x be a solution of (1)–(5). Then, by the flow conservation constraints (3), x is the incidence vector of a union of r dicycles $C_1, \dots, C_r$ and each $C_i \subseteq A_{k_i}$ for some $k_i \in M$. Because of (1) there is exactly one dicycle $C_i \subseteq A_k$ for each $k \in M$ with $0 \in V_k(C_i)$. Likewise, by (2), for each $v \in J$ there is exactly one dicycle C_i with $v \in V_{k_i}(C_i)$. Furthermore, for each C_i we must have $0 \in V_{k_i}(C_i)$, because otherwise $S = V_{k_i}(C_i)$ would lead to a violated $v0$ -cut inequality (4) for $k = k_i$ and arbitrary $v \in S$. Thus, $r = m$ and the cycles $C_1, \dots, C_m$ fulfill the requirements for $\mathcal{F}_n^m$. $\square$

So P_n^m is the convex hull of the feasible solutions of (1)–(5). In order to determine the dimension of P_n^m we investigate equations (1)–(3) which we collect, for convenience, in a matrix B and a right hand side vector b ,

$$Bx = b.$$

Lemma 2. *The coefficient matrix $B \in \{-1, 0, 1\}^{(m+n+mn) \times |A|}$ corresponding to (1)–(3) has full row rank $m + n + mn$.*

Proof. Consider the sets of column indices

$$\begin{aligned} F_1 &= \{0i1, i01 : i \in J\} \cup \{121\} \\ F_k &= \{00k\} \cup \{0ik : i \in J\} \quad \text{for } k \in M \setminus \{1\} \\ F &= \bigcup_{k \in M} F_k. \end{aligned}$$

We claim that the columns of B corresponding to F are linearly independent.

The linear independence of F_h for $h \in M \setminus \{1\}$ with respect to $F \setminus F_h$ follows from the fact that the variables are the only arcs in F that are in the support of the equation with $k = h$ in (1) and of the n equations with $k = h$ in (3). A proper arrangement of the submatrix of B consisting of these rows and the columns corresponding to F_h yields an upper triangular matrix with all diagonal entries equal to one. Thus the columns corresponding to F_h are linearly independent.

In the case of F_1 variables $0i1$ and $i01$ are both in the support of (3) for $k = 1$ and $j = i$ but only $i01$ is in (2) for $j = i$, so they are linearly independent with respect to these rows and the rows do not contain further variables in $F \setminus \{121\}$. Now observe that column 001 is the linear combination (of columns) $021 + 101 - 121$. Variable 001 is only in the support of the equation with $k = 1$ in (1) and its column is therefore not in the span of the columns of $F \setminus \{121\}$. We conclude that the columns of F are linearly independent. $\square$

To complement this upper bound on the dimension of P_n^m with a lower bound we will need a standard construction of linearly independent tours for the ATSP.

Theorem 1. *(see [10]) Let D_V be the complete digraph on nodes $V = \{0, \dots, n\}$ with $n \geq 2$. There exist $n(n-1)$ tours in D_V that are linearly independent with respect to the arcs not incident to 0.*

Proof. Follows directly from [10], proof 1 of Theorem 7 and Theorem 20. $\square$

Now we are ready to determine the dimension of P_n^m .

Theorem 2. $\dim(P_n^m) = mn^2 - n$ for $n \geq 3$ and $m \geq 2$.

Proof. It follows from Lemma 2 that $\dim(P_n^m) \leq mn^2 - n$. To prove equality let $a^T x = a_0$ be a valid equation for P_n^m . We will show that it is linearly dependent with respect to $Bx = b$. By the proof of Lemma 2 the columns of B corresponding to the arcs in $F = \bigcup_{k \in M} F_k$ are linearly independent, so we can compute $\lambda \in \mathbb{Q}^{n+m+nm}$ such that $a_{ijk}^T = (\lambda^T B)_{ijk}$ for $ijk \in F$. The equation $d^T x = (a^T - \lambda^T B)x = a_0 - \lambda^T b$ is again valid for P_n^m and $d_{ijk} = 0$ for $ijk \in F$. We will show that all other coefficients d_{ijk} have to be zero as well.

First observe that on the support $A_1 \setminus \{001\}$ d has exactly $n(n-1) - 1$ ‘unknowns’. Employing the $n(n-1)$ linearly independent tours of Lemma 1 construct linearly independent feasible solutions $A^h \in \mathcal{F}_n^m$ for $h = 1, \dots, n(n-1)$ that use these tours on machine 1 such that the x^{A^h} are linearly independent on the index set $\{ij1 : i \neq j, i, j \in J\}$. There exists an $\hat{h}$, $1 \leq \hat{h} \leq n(n-1)$, such that the $n(n-1) - 1$ vectors $x^{A^h} - x^{A^{\hat{h}}}$ ($h \neq \hat{h}$) are linearly independent on the index set $\{ij1 : i \neq j, i, j \in J\} \setminus \{121\}$. These difference vectors have no support outside $A_1 \setminus \{001\}$ and satisfy $d^T(x^{A^h} - x^{A^{\hat{h}}}) = 0$. Collect the $n(n-1) - 1$ difference vectors as columns in a matrix X . Since $d^T X = 0$ and $d_{ij1} = 0$ for $ij1 \in F_1$ we conclude that $d_{ij1} = 0$ for all $ij1 \in A_1 \setminus \{001\}$ by the linear independence of the columns of X .

We proceed to show that $d_{ij2} = 0$ for all $ij2 \in A_2$. The following two arc sets $A^1, A^2 \in \mathcal{F}_n^m$ correspond to a tour on machine 1 and to the same tour shortened by ‘moving’ node n to machine 2,

$$\begin{aligned} A^1 &= \{i(i+1)1 : i = 0, \dots, n-1\} \cup \{n01\} \cup \{00k : 2 \leq k \in M\} \\ A^2 &= \{i(i+1)1 : i = 0, \dots, n-2\} \cup \{(n-1)01\} \cup \{0n2, n02\} \\ &\quad \cup \{00k : 2 < k \in M\}. \end{aligned}$$

Since $0 = d^T(x^{A^1} - x^{A^2}) = -d_{n02}$ (in the support of $x^{A^1} - x^{A^2}$ only d_{n02} has not yet been shown to be zero) we obtain $d_{n02} = 0$. Moving any other vertex $i \in J$ from the tour on machine 1 to machine 2 yields $d_{i02} = 0$. Now it is easy to obtain $d_{ij2} = 0$ for $i \neq j$ and $i, j \in J$ by the dicycles $\{0i2, ij2, j02\}$ (here we use $n \geq 3$ as otherwise d_{001} would have to enter). So $d_{ij2} = 0$ for $ij2 \in A_2$. The same steps for $k > 2$ lead to $d_{ijk} = 0$ for $ijk \in A_k$ for $2 \leq k \in M$.

It remains to show that $d_{001} = 0$. This is easily achieved by taking the difference vector between x^{A^1} and an incidence vector of a feasible arc set $A^3 \in \mathcal{F}_n^m$ with $001 \in A^3$ (e.g., let A^3 contain a tour on machine 2). $\square$

Remark 1. It can be shown that for $n = 2$ and $m \geq 2$ the dimension is $mn^2 - n - 1$, the additional affine constraint being

$$\sum_{ijk \in \{01k, 10k, 12k, 21k : k \in M\}} x_{ijk} = 2.$$

Before we turn to the facets of P_n^m we study a tightly related problem.

4 The One-Machine Subproblem

Consider the subproblem on one machine where this machine is not required to process all jobs, but may process any subset of jobs, or even choose the idle loop. We model this problem on a digraph $D_s = (V_s, A_s)$ with $V_s = \{0\} \cup J$ and $A_s = \{(0, 0)\} \cup \{(i, j) : i, j \in V_s, i \neq j\}$. The feasible set is

$$\mathcal{F}_n^s = \{C : C \subseteq A_s \text{ is a dicycle with } 0 \in V_s(C)\}$$

The corresponding polytope defined by the incidence vectors of the arc sets in the feasible set is

$$P_n^s = \text{conv} \{x^A : A \in \mathcal{F}_n^s\}$$

We can model the incidence vectors by the following constraints,

$$x_{00} + x(\delta^+(0)) = 1 \tag{6}$$

$$x(\delta^-(j)) = x(\delta^+(j)) \quad j \in J \tag{7}$$

$$x(\delta^+(S)) \geq x(\delta^+(v)) \quad S \subseteq J, |S| \geq 2, v \in S \tag{8}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in A_s. \tag{9}$$

Lemma 3. *An incidence vector x^A of $A \subset A_s$ is a feasible solution of (6)–(9) if and only if $A \in \mathcal{F}_n^s$.*

Proof. The *if* part is easy to check, so consider the *only if* part. Let x be a solution of (6)–(9). Then, by the flow conservation constraints (7), x is the incidence vector of a union of r dicycles $C_1, \dots, C_r$. Because of (6) exactly one dicycle of these, w.l.o.g. C_1 , satisfies $0 \in V(C_1)$. Furthermore, for all C_i we must have $0 \in V(C_i)$, because otherwise $S = V(C_i)$ would lead to a violated $v0$ -cut inequality (8) with arbitrary $v \in S$. Thus $r = 1$ and the cycles C_1 fulfill the requirements for $\mathcal{F}_n^s$. $\square$

P_n^s is therefore the convex hull of the feasible solutions of (6)–(9). We determine the dimension of P_n^s by the same steps as in Section 3. We collect equations (6) and (7) in a matrix B_s and a right hand side vector b_s , $B_s x = b_s$.

Lemma 4. *The coefficient matrix $B_s \in \{-1, 0, 1\}^{(n+1) \times |A_s|}$ corresponding to (6) and (7) has full row rank $n + 1$.*

Proof. The set of column indices

$$F_s = \{00\} \cup \{0i : i \in J\}$$

corresponds to the sets F_k for $k > 1$ of the proof of Lemma 2. The proof can be completed analogous to the proof there. $\square$

Theorem 3. $\dim(P_n^s) = n^2$.

Proof. $\dim(P_n^s) \leq n^2$ follows from Lemma 4. We construct $n^2 + 1$ feasible dicycles,

$$\begin{aligned} C_{00} &= \{\mathbf{00}\} \\ C_{i0} &= \{0i, \mathbf{i0}\} & i \in J \\ C_{ij} &= \{0i, \mathbf{ij}, j0\} & i, j \in J, i \neq j. \end{aligned}$$

To show the linear independence of the incidence vectors $x^{C_{ij}}$ arrange them as rows in a matrix B . Let F be the set of indices indicated in bold above, $F = \{00\} \cup \{i0 : i \in J\} \cup \{ij : i, j \in J, i \neq j\}$, and consider the submatrix $B_s \in \{0, 1\}^{|F| \times |F|}$ consisting of the columns corresponding to F . Reorder rows and columns of B_s simultaneously arranging rows and columns corresponding to $\{ij : i, j \in J, i \neq j\}$ in front, followed by rows and columns $\{00\} \cup \{i0 : i \in J\}$. The reordered matrix is upper triangular with all diagonal elements equal to one. Thus the vectors are linearly independent. $\square$

Theorem 4. For $ij \in A_s$, $x_{ij} \geq 0$ defines a facet of P_n^s if $n \geq 3$.

Proof. For 00 or $ij \in A_s$ with $i, j \in J$, $i \neq j$ we have to drop dicycle C_{ij} from the dicycles defined in the proof of Theorem 3 to obtain n^2 linearly independent tours satisfying $x_{ij} = 0$. (Indeed, these are facets for $n \geq 1$.)

For $0v \in A_s$ with $v \in J$ we assume w.l.o.g. that $v = 1$. Then the n^2 feasible dicycles

$$\begin{aligned} C_{00} &= \{\mathbf{00}\} \\ C_{0i} &= \{0i, \mathbf{i0}\} & i \in J \setminus \{1\} \\ C_{ij} &= \{0i, \mathbf{ij}, j0\} & i \in J \setminus \{1\}, j \in J, i \neq j \\ C_{1j} &= \{0i, i1, \mathbf{1j}, j0\} & j \in J \setminus \{1\} \text{ with some } i \in J \setminus \{1, j\} \end{aligned}$$

are again easily seen to satisfy $x_{01} = 0$ and to be linearly independent (proceed as in the proof of Theorem 3). A similar construction shows that x_{v0} is facet defining for $v \in J$. $\square$

The following corollary will be especially useful in relating P_n^s with P_n^m .

Corollary 1. Any facet defining inequality $a^T x \leq a_0$ of P_n^s not equivalent to $x_{00} \geq 0$ satisfies $a^T x^{C_{00}} = a_0$ where $C_{00} = \{00\}$.

Proof. Let $a^T x \leq a_0$ be facet defining with $C_{00} \notin \mathcal{F}_a = \{A \in \mathcal{F}_n^s : a^T x^A = a_0\}$. Since $A = C_{00}$ is the only set $A \in \mathcal{F}_n^s$ satisfying $x_{00}^A = 1$ it follows that $\mathcal{F}_a \subset \{A \in \mathcal{F}_n^s : x_{00}^A = 0\}$, so $a^T x \leq a_0$ is equivalent to the facet $x_{00} \geq 0$. $\square$

Theorem 5. For $S \subseteq J$, $|S| \geq 2$, $v \in S$ inequalities $x(\delta^+(S)) \geq x(\delta^+(v))$ are facet defining for P_n^s .

Proof. W.l.o.g. let $S = \{1, \dots, h\}$ with $1 < h \leq n$ and $v = 1$. For convenience, let $a^T x \geq 0$ denote the inequality $x(\delta^+(S)) \geq x(\delta^+(1))$, and $S_1 = S \setminus \{1\}$. We list n^2 feasible dicycles whose incidence vectors are linearly independent and satisfy $a^T x^{C_i} = 0$:

1. $C_{00} = \{\mathbf{00}\}$; the ‘idle’ loop.
2. $C_{i0} = \{0i, \mathbf{i0}\}$ for $i \in J \setminus S_1$; $n - h + 1$ dicycles.
3. $C_{ij} = \{0i, \mathbf{ij}, j0\}$ for $i, j \in J \setminus S_1, i \neq j$; $(n - h + 1)(n - h)$ dicycles.
4. $C_{i1} = \{0i, \mathbf{i1}, 10\}$ for $i \in S_1$; $h - 1$ dicycles.
5. $C_{ij} = \{0i, \mathbf{ij}, j1, 10\}$ for $i \in J \setminus \{1\}, j \in S \setminus \{1, i\}$; $(n - 2)(h - 1)$ dicycles.
6. $C_{ij} = \{01, 1i, \mathbf{ij}, j0\}$ for $i \in S_1, j \in J \setminus S$; $(n - h)(h - 1)$ dicycles.
7. $C_{i0} = \{01, 1i, \mathbf{i0}\}$ for $i \in S_1$; $h - 1$ dicycles.
8. $C_{1i} = \{01, \mathbf{1i}, i(i + 1), (i + 1)0\}$ for $i \in \{2, \dots, h - 1\}$; $h - 2$ dicycles.

To see the linear independence, collect the incidence vectors of the dicycles as rows in a matrix. By arranging the rows corresponding to 1–6 in reverse order and the columns of the variables indicated in bold ‘in front on the diagonal’ then this submatrix is upper triangular with each diagonal element equal to 1.

On the other hand the submatrix corresponding to the rows of 7 and 8 is regular on the support indicated in bold (on this support the submatrix is the edge-node incidence matrix of an undirected path). The only other type of rows having these variables in their support is in 6, but the rows of 6 are already determined uniquely by their support in bold, which is not in the support of any other row. So 6–8 are linearly independent on their support in bold and none of the variables in this support appear in any other row, therefore the incidence vectors of the cycles 1–8 are linearly independent. $\square$

It is to be expected that the following theorem has already been proved for the PCTSP, but we could not find an explicit reference so far.

Theorem 6. *For $x \in \mathbb{Q}^{n(n+1)+1}$, $x \geq 0$ satisfying (7), inequalities (8) can be separated in polynomial time.*

Proof. For each node $v \in J$ construct a digraph $D_v = (V_s, A_v)$ where $A_v = \{ij : i \in J, j \in V_s \setminus \{v\}\}$ and define capacities $c_{ij}^v = x_{ij}$ for $ij \in A_v$. Solve, in polynomial time, a v0-max flow problem for these capacities to obtain a min-cut set $\delta_+(S_v)$ with $v \in S_v \subset J$ for these capacities. If $x(\delta_+(S_v)) < x(\delta^+(v))$ then a violated inequality has been found, otherwise (8) is satisfied for all $S \subset J$ with $v \in S$. $\square$

In order to show the relation between P_n^s and the polytope P_0 for the PCTSP as defined in [1,2] we introduce loop variables x_{jj} (called y_j in [2]) and add to (6)–(9) the constraints

$$x_{jj} + x(\delta^-(j)) = 1 \quad j \in J. \quad (10)$$

The polyhedron $P_n^s(n)$ corresponding to the convex hull of the points satisfying constraints (6)–(9) and (10) is not exactly P_0 of [2], but the only differences are that now the ‘long’ cycle is required to pass through node 0 and that this long cycle may also be of length 1 as opposed to at least two. This close relationship allows to copy the lifting Theorem 3.1 of [2] more or less verbatim. Denote by $P_n^s(k)$ the polytope obtained from the ATSP polytope P (with respect to the complete digraph on nodes $J \cup \{0\}$) by introducing the first k loop variables x_{jj} for $j = 0, \dots, k$.

Theorem 7. Let $a^T x \leq a_0$ be any facet defining inequality for the ATSP polytope P with respect to the complete digraph on V_s . For $k = 0, \dots, n$ define $b_{j_k} = a_0 - z(P_n^s(k))$, where

$$z(P_n^s(k)) = \max \left\{ a^T x + \sum_{i=0}^{k-1} b_{j_i} x_{j_i j_i} : x \in P_n^s(k), x_{j_k j_k} = 1 \right\}.$$

Then the inequality $a^T x + \sum_{i=0}^{k-1} b_{j_i} x_{j_i j_i} \leq a_0$ is valid and facet defining for $P_n^s(k)$.

Proof. As in [2], but with $P_0(k)$ replaced with $P_n^s(k)$. □

After having determined the lifting coefficients we eliminate variables x_{jj} for $j \in J$ by using (10) in order to obtain the lifted inequality for P_n^s .

For example the v0-cut inequalities can be seen to be liftings of the subtour elimination constraints for $0 \notin S \subset V_s$, $|S| \geq 2$, with respect to the sequence $j_1, \dots, j_{|S|} = v \in S$ and afterwards for $j \in V_s \setminus S$. Here, we need the direct proof of Theorem 5 in order to arrive at Theorem 11.

It is not yet clear for which other classes of facet defining inequalities of the ATSP polytope it is possible to determine the lifting coefficients explicitly as in [2], because the special role of node 0 does not allow to use these results without some careful checking.

For the following class of facet defining inequalities we do not know whether there exists a corresponding class in the ATSP or PCTSP literature.

Theorem 8. Let $S_0 \subset S_1 \subset \dots \subset S_k = J$ be a nested sequence of proper subsets with $k \geq 2$ and let $t_0 \in S_0$, $t_i \in S_i \setminus S_{i-1}$ for $i = 1, \dots, k$. For

$$A_c = \bigcup_{i=1, \dots, k} [\delta^+(S_{i-1}) \setminus \{vt_i : v \in S_{i-1}\}]$$

the nested conflict inequality

$$\sum_{i=1}^k x_{t_i t_0} - \sum_{ij \in A_c} x_{ij} \leq 0 \tag{11}$$

is facet defining for P_n^s if $n \geq 3$.

Proof. Validity: Any dicycle through 0 contains at most one of the arcs $t_i t_0$ with $1 \leq i \leq k$. If it does so for some i then it must include an arc from the cut set $\delta^+(S_{i-1}) \setminus \{vt_i : v \in S_{i-1}\}$ in order to return from the set S_{i-1} to 0 without visiting t_i once more.

Facet defining: We exhibit n^2 linearly independent dicycles through 0 that satisfy (11) with equality. In order to simplify notation we will only give the sequence of vertices of each dicycle, consecutive multiple appearances of the same vertex should be interpreted as a single appearance, a sequence $t_j, t_{j+1}, \dots, t_k$ is regarded as empty if $j > k$. Furthermore, we will use $\Delta_i = S_i \setminus S_{i-1} \setminus \{t_i\}$ for $1 \leq i \leq k$ and $\Delta_0 = S_0 \setminus \{t_0\}$ (these sets may be empty).

1. The ‘idle’ loop: $0, 0$
2. $v \in \Delta_i, w \in \Delta_j, 0 \leq i < j \leq k: 0, t_j, t_0, t_1, \dots, t_i, v, w, t_{j+1}, t_{j+2}, \dots, t_k, 0$
3. $v \in \Delta_i, w \in \Delta_j, k \geq i \geq j \geq 0, v \neq w: 0, v, w, t_{j+1}, \dots, t_k, 0$
4. $v \in \Delta_i, w = t_j, 0 \leq i < j - 1 \leq k - 1: 0, t_{j-1}, t_0, t_1, \dots, t_i, v, t_j, t_{j+1}, \dots, t_k, 0$
5. $v = t_i, w \in \Delta_j, 0 \leq i < j \leq k: 0, t_j, t_0, t_1, \dots, t_i, w, t_{j+1}, t_{j+2}, \dots, t_k, 0$
6. $v = t_i, w \in \Delta_0: 0, t_k, t_0, t_1, w, 0$
7. $v \in \Delta_i, w = 0, 0 \leq i \leq k - 1: 0, t_k, t_0, t_1, \dots, t_i, v, 0$
8. $v = t_i, w \in \Delta_j, k \geq i \geq j \geq 0, i > 1: 0, t_i, w, t_1, t_0, 0$
9. $v = t_i, w \in \Delta_i, 0 \leq i \leq 1: 0, t_i, w, t_{i+1}, t_{i+2}, \dots, t_k, 0$
10. $v \in \Delta_i, w = t_j, k \geq i \geq 0, i + 1 \geq j, k \geq j \geq 0: 0, v, t_j, t_{j+1}, \dots, t_k, 0$
11. $v \in \Delta_k, w = 0: 0, v, 0$
12. $v = t_i, w = 0, 1 \leq i \leq k - 1: 0, t_{i+1}, t_0, t_1, \dots, t_i, 0$
13. $v = t_i, w = t_j, 0 \leq i < j - 1 \leq k - 1: 0, t_{i+1}, t_0, t_1, \dots, t_i, t_j, t_{j+1}, \dots, t_k, 0$
14. $v = t_i, w = t_{i+1}, 0 \leq i \leq k - 1: 0, t_i, t_{i+1}, \dots, t_k, 0$
15. $v = t_i, w = t_j, k \geq i > j \geq 0: 0, t_i, t_j, t_0, 0$
16. $v = t_k, w = 0: 0, t_k, 0$

For each edge $vw \in A_s \setminus \{01, \dots, 0n, t_00\}$ (in total n^2 edges) the list provides an associated dicycle via v and w as specified above. Observe that an identifying edge in this list does not appear in dicycles listed after the specification of its associated dicycle (an appropriate internal order can be defined for 14 and 15). Thus, by arranging the incidence vectors of the dicycles as rows and the edges as columns according to this ordering the resulting matrix is upper triangular with ones on the main diagonal and therefore the vectors are linearly independent. $\square$

5 Facets of the m-Cost ATSP Polytope

Theorem 9. *For $ijk \in A$, $x_{ijk} \geq 0$ defines a facet of P_n^m if $n \geq 3$ and $m \geq 2$.*

Proof. Without loss of generality let $k = 2$.

For 002 or $ij2 \in A$ with $i, j \in J$ we first construct $n(n-1)$ tours on machine 1 as in the proof of Theorem 2. Then we construct n^2 solutions corresponding to the dicycles given in the proof of Theorem 4, where these dicycles are used on machine 2, whereas the remaining nodes of J are covered by dicycles on machine 1. Finally we construct a tour on machine 2 without using the current arc in question, which is the first solution containing 001 in its support. If $m > 2$ further linear independent solutions can be constructed by using the dicycles of the proof of Theorem 3 on these machines and by completing them on machine 1.

For indices $0v2$ we show this, w.l.o.g., for $v = 1$. The proof is analogous, it uses the corresponding dicycles of Theorem 4. The final tour for exploiting 001 has the form $\{0n2\} \cup \{i(i-1)2 : i \in J\}$. Since for $n = 3$ this tour does not have common support with other tours of Theorem 4 with respect to their support indicated in bold, it is linearly dependent with respect to dicycles only, that do not have 001 in their support. $\square$

Theorem 10. *For $m \geq 3$ and $n \geq 3$ all facet defining inequalities $\hat{a}^T x \leq \hat{a}_0$ of P_n^s give rise to facet defining inequalities $a^T x \leq a_0$ for P_n^m by identifying A_s with A_h for some $h \in M$, i.e., by setting $a_{ijh} = \hat{a}_{ij}$ for $ij \in A_s$, $a_{ijk} = 0$ for $ijk \in \mathcal{A} \setminus A_h$, and $a_0 = \hat{a}_0$.*

Proof. W.l.o.g. let $h = m$. It follows from Theorem 9 that $x_{00m} \geq 0$ is facet defining. So let $\hat{a}^T y \leq \hat{a}_0$ be a facet defining inequality of P_n^s not equivalent to $x_{00} \geq 0$. Then Corollary 1 guarantees that any $x^{A^{m-1}}$ with $A^{m-1} \in \mathcal{F}_n^{m-1}$ is easily extended to a solution x^{A^m} with $A^m \in \mathcal{F}_n^m$ satisfying $a^T x^{A^m} = a^0$ by setting $A^m = A^{m-1} \cup \{00m\}$. By Theorem 2 this ensures the existence of $(m-1)n^2 - n + 1$ solutions that are linearly independent on the support outside A_m . Since $\hat{a}^T x \leq \hat{a}_0$ is a facet of P_n^s we can construct further $n^2 - 1$ linearly independent solutions with respect to the support in $A_m \setminus \{00m\}$ by completing the corresponding solutions of $\mathcal{F}_n^s$ on machine 1 to solutions of $\mathcal{F}_n^m$. $\square$

We do not know whether the theorem also holds for $m = 2$. The main difficulty is that one has to construct an additional solution that exploits the variable 001. We can do this for the $v0$ -cut and the nested conflict inequalities.

Theorem 11. *The $v0$ -cut inequalities (4) are facet defining for P_n^m for $m \geq 2$ and $n \geq 3$.*

Proof. By Theorem 10 we may concentrate on the case $m = 2$. W.l.o.g. consider in (4) the case $k = 2$, $S = \{1, \dots, h\}$ with $1 < h \leq n$ and $v = 1$. For convenience, let $a^T x \geq 0$ denote the inequality $x(\delta_2^+(S)) \geq x(\delta_2^+(1))$, and $S_1 = S \setminus \{1\}$.

We use $n(n-1)$ tours on machine 1 as in the proof of Theorem 2 and $n^2 - 1$ additional solutions corresponding to the dicycles defined in the proof of Theorem 5 on machine 2, completed to full solutions on machine 1. These $2n^2 - n - 1$ solutions are linearly independent and satisfy $a^T x = 0$. If $n > 3$ then a tour on machine 2 suffices, since all other solutions have $x_{001} = 0$. If $n = 3$ then consider the tour $C = \{022, 212, 132, 302\}$. Eliminate in its incidence vector the support in 212 by subtracting an incidence vector of type 4 (in combination with the type 2 incidence vector $\{012, 102\}$) and eliminate the support in 132 as well as 302 by subtracting an incidence vector of either type 3 or 7, then all relevant support on machine 2 has been eliminated while 001 is still in the support. So the tour is linearly independent with respect to all other solutions. $\square$

Theorem 12. *For $x \in \mathbb{Q}^{|\mathcal{A}|}$, $x \geq 0$ satisfying (3), the $v0$ -cut inequalities (4) can be separated in polynomial time.*

Proof. Follows directly from Theorem 6. $\square$

Theorem 13. *The nested conflict inequalities (11) are facet defining for P_n^m for $m \geq 2$ and $n \geq 3$.*

Proof. By Theorem 10 we may concentrate on the case $m = 2$. Let k, S_i, t_i, Δ_i be defined as in Theorem 8 and its proof and identify the edges, w.l.o.g., with the edges of machine 2. Denote by $h_i = |\Delta_i|$ and let $\Delta_i = \{v_1^i, \dots, v_{h_i}^i\}$, the sets may be empty. Consider the tour on machine 2 specified by the sequence of vertices

$$0, v_1^k, v_2^k, \dots, v_{h_k}^k, v_1^{k-1}, \dots, v_{h_{k-1}}^{k-1}, v_1^{k-2}, \dots, v_{h_0}^0, t_1, t_2, \dots, t_k, t_0, 0.$$

On the support $\mathcal{A} \setminus A_1 \setminus \{012, \dots, 0n2, t_0 02\}$ the incidence vector (with respect to $\mathcal{A}$) of this tour is a linear combination of incidence vectors (with respect to $\mathcal{A}$) of dicycles from 3, 10, 14, and the dicycle $0, t_k, t_0, 0$ of 15 as specified in the proof of Theorem 8. None of these dicycles are tours on machine 2, so 001 is not in the support of any of these incidence vectors. However, 001 is in the support of the new tour. Thus, by the linear independence of the dicycles of the proof of Theorem 8, its incidence vector is linearly independent with respect to the $n(n-1)$ tours on machine 1 of the proof of Theorem 2 and the n^2-1 dicycles (the idle dicycle is covered by the tours on machine 1) of the proof of Theorem 8. $\square$

6 Subtour Elimination Constraints

We consider the standard transformation from m-ATSP (with identical machines) to standard ATSP (see e.g. [16]). To this end let $D = (V, \mathcal{B})$ be a digraph with $V = \{0_1, \dots, 0_m\} \cup J$ and

$$\begin{aligned} \mathcal{B} = & \{0_k i, i 0_k : k \in M, i \in J\} \cup \\ & \{ij : i \neq j, i, j \in J\} \cup \\ & \{0_k, 0_{k+1} : k \in M \setminus \{m\}\} \cup \\ & \{0_m 0_1\}. \end{aligned}$$

We map vectors $x \in \mathbb{Q}^{|\mathcal{A}|}$ to $Tx = y \in \mathbb{Q}^{|\mathcal{B}|}$ by

$$\begin{aligned} y_{0_k i} &= x_{0ik} & i \in J, k \in M \\ y_{i 0_1} &= x_{i 0m} & i \in J \\ y_{i 0_k} &= x_{i 0(k-1)} & i \in J, k \in \{2, \dots, M\} \\ y_{ij} &= \sum_{k \in M} x_{ijk} & i \neq j, i, j \in J \\ y_{0_k 0_{k+1}} &= x_{00k} & k \in \{1, \dots, m-1\} \\ y_{0_m 0_1} &= x_{00m} \end{aligned}$$

It is well known that any feasible solution x^A with $A \in \mathcal{F}_n^m$ maps to $y^B = Tx^A$ with B being a tour in D . Probably the most useful inequalities for the general ATSP are the subtour elimination constraints, usually stated in the form

$$y(\mathcal{B}(S)) \leq |S| - 1 \quad \text{for } S \subset V, 2 \leq |S| \leq |V| - 2.$$

In order to show that Tx satisfies all subtour elimination constraints if x satisfies all v0-cut inequalities, we need the following observation.

Proposition 2. *Let $x \in \mathbb{Q}^{|\mathcal{A}|}$, $x \geq 0$, satisfy constraints (1) to (3). If for some $S \subset V$, $2 \leq |S| \leq |V| - 2$, there exist $h, k \in M$, $h \neq k$ such that $0_h \in S$ and $0_k \notin S$ then the subtour elimination constraint corresponding to S cannot be violated for $y = Tx$.*

Proof. W.l.o.g. assume $h = 1, k = 2$ and let $x \in \mathbb{Q}^{|\mathcal{A}|}$, $x \geq 0$. Decompose the flow through h into cycles C_1 to C_r . In the transformed graph each of these cycles correspond to a path from 0_1 to 0_2 and must therefore cross $\delta^+(S)$. Therefore, $1 = x_{001} + x(\delta_1^+(0)) \leq y(\delta^+(S))$, which implies because of $y(\delta^+(S)) + y(\mathcal{B}(S)) = \sum_{v \in S} y(\delta^+(v)) = |S|$ that $y(\mathcal{B}(S)) \leq |S| - 1$. $\square$

Theorem 14. *Let $x \in \mathbb{Q}^{|\mathcal{A}|}$, $x \geq 0$, satisfy constraints (1) to (4), let $y = Tx$, and let $S \subset V$ with $2 \leq |S| \leq |V| - 2$. Then $y(\mathcal{B}(S)) \leq |S| - 1$.*

Proof. Because of Proposition 2 we may assume $0_k \notin S$ for $k \in M$. First observe that

$$\sum_{v \in S} x(\delta_k^+(S)) - \sum_{v \in S} x(\delta_k^+(v)) = |S|x(\delta_k^+(S)) - x(\delta_k^+(S)) - x(A_k(S)).$$

By summing constraints (4) over all $k \in M$ and $v \in S$ we obtain

$$\begin{aligned} 0 &\leq \sum_{k \in M} \sum_{v \in S} [x(\delta_k^+(S)) - x(\delta_k^+(v))] = \\ &= \sum_{k \in M} [|S|x(\delta_k^+(S)) - x(\delta_k^+(S)) - x(A_k(S))] \\ &= (|S| - 1) \sum_{k \in M} x(\delta_k^+(S)) - \sum_{k \in M} x(A_k(S)) \\ &= (|S| - 1)y(\delta^+(S)) - y(\mathcal{B}(S)) \\ &= (|S| - 1)y(\delta^+(S)) + (|S| - 1)[y(\mathcal{B}(S)) - y(\mathcal{B}(S))] - y(\mathcal{B}(S)) \\ &= (|S| - 1) \sum_{v \in S} y(\delta^+(v)) - |S|y(\mathcal{B}(S)) \\ &= (|S| - 1) \sum_{v \in S} 1 - |S|y(\mathcal{B}(S)) = (|S| - 1)|S| - |S|y(\mathcal{B}(S)) \end{aligned}$$

It follows that $y(\mathcal{B}(S)) \leq |S| - 1$ and the proof is complete. $\square$

7 Numerical Results

The scheduling problem at Herlitz PBS AG asks for a solution minimizing the makespan over two non-identical machines. Conflicts arising from jobs using the same printing cylinders are avoided by additional side constraints that require that these jobs be executed on the same machine. The problem is thus not a pure m-Cost ATSP, but the basic structure is the same. Our data instances

are constructed by randomly choosing jobs (without repetitions) from a set of roughly 150 jobs provided by Herlitz PBS AG. The cost coefficients are computed according to data and rather involved rules determined in cooperation with experts at Herlitz PBS AG. The cost of one arc c_{ijk} includes the setup time from job i to job j and the printing time of job j on machine k .

Our computational experiments were performed on a Sun Ultra 10 with a 299 MHz SUNW,UltraSPARC-III CPU and with CPLEX 6.0 as LP-solver, computation times are given in seconds. Table 1 shows detailed results for 10 random instances of 60 jobs, the typical number of jobs processed in one week.

Table 1. 10 instances of 60 jobs each

n	feas	v0-cut (%)	secs	subtour (%)	subt+ D_k (%)
60	9420	9406.8 (0.14)	19.6	9387.42 (0.35)	9387.42 (0.35)
60	9382	9366.93 (0.16)	10.5	9345.37 (0.39)	9345.48 (0.39)
60	9519	9481.32 (0.4)	6.8	9425.49 (0.98)	9425.49 (0.98)
60	9723	9707.24 (0.16)	12.2	9674.58 (0.5)	9674.58 (0.5)
60	9910	9900.59 (0.1)	31.4	9859.52 (0.51)	9860.24 (0.5)
60	9421	9411.46 (0.1)	8.5	9395.92 (0.27)	9395.92 (0.27)
60	9984	9967.58 (0.16)	14.4	9914.74 (0.69)	9915.24 (0.69)
60	9557	9542.84 (0.15)	17.9	9517.14 (0.42)	9517.14 (0.42)
60	9263	9261.73 (0.01)	7.4	9238.88 (0.26)	9238.88 (0.26)
60	9373	9343.13 (0.32)	23.9	9294.81 (0.83)	9294.81 (0.83)

Column n gives the size of the instance, $feas$ displays the best integral solution we know, $v0-cut$ shows the optimal value of the relaxation based on the v0-cut inequalities, in parenthesis we give the relative gap $1 - (v0-cut/feas)$. $secs$ is the number of seconds the code needed to solve this relaxation, including separation. We compare the value of relaxation $v0-cut$ to the relaxations obtained by separating the subtour inequalities on the transformed problem and to the relaxation combining subtour and D_k^+/D_k^- inequalities (for the definition of D_k^+/D_k^- inequalities see [10], for a separation heuristic [7]). With respect to the latter relaxations the v0-cut inequalities close the gap by roughly 60%. Optimal solutions to the subtour relaxations without v0-cut inequalities would typically exhibit several subtours on the one machine subproblems. There is still much room for improvement in the separation of the v0-cut inequalities, so it should be possible to reduce the computation time in more sophisticated implementations. It is surprising that the D_k^+/D_k^- inequalities are not very effective, quite contrary to the experience in usual ATSP-problems.

In Table 2 we summarize results on instances with 40 to 100 jobs. For each problem size we generated 10 instances, the table displays the average relative gap of the relaxations as well as the average computation time. In the application in question the error in the data is certainly significantly larger than 0.2%, so there is no immediate need to improve these solutions further. On the other

Table 2. Relative gap and computation time, averaged over 10 instances

n	#	v0-cut %	secs	subtour %	subt+ D_k %
40	10	0.22	2.84	0.54	0.54
50	10	0.15	9.57	0.51	0.51
60	10	0.17	15.25	0.52	0.52
70	10	0.16	38.13	0.52	0.52
80	10	0.18	55.19	0.49	0.48
90	10	0.23	82.75	0.54	0.54
100	10	0.19	135.91	0.44	0.44

hand, most fractional solutions we investigated so far exhibited violated nested conflict inequalities, so separation heuristics for this new class of inequalities might help to improve the performance.

I thank Norbert Ascheuer for many helpful discussions and pointers to the literature and software, Oleg Mänz for helping in the implementation of the v0-cut separator, and Thorsten Koch for making available his min-cut code.

References

1. E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.
2. E. Balas. The prize collecting traveling salesman problem: II. polyhedral results. *Networks*, 25:199–216, 1995.
3. E. Balas and M. Fischetti. A lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets. *Mathematical Programming*, 58:325–352, 1993.
4. M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors. *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*. Elsevier Sci. B.V., Amsterdam, 1995.
5. J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. *Time Constrained Routing and Scheduling*, chapter 2, pages 35–139. Volume 8 of Ball et al. [4], 1995.
6. M. Fischetti, J. Salazar González, and P. Toth. A branch-and-cut algorithm for the generalized travelling salesman problem. Technical report, University of Padova, 1994.
7. M. Fischetti and P. Toth. A polyhedral approach to the asymmetric traveling salesman problem. *Management Science*, 43:1520–1536, 1997.
8. M. Fischetti and P. Toth. An additive approach for the optimal solution of the prize-collecting travelling salesman problem. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 319–343. Elsevier Science Publishers B.V. (North-Holland), 1998.
9. M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32:263–273, 1998.
10. M. Grötschel and M. Padberg. Polyhedral theory. In Lawler et al. [12], chapter 8.
11. M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, chapter 4, pages 225–330. North Holland, 1995.

12. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. John Wiley & Sons Ltd, Chichester, 1985.
13. M. Padberg and M. Grötschel. Polyhedral computations. In Lawler et al. [12], chapter 9.
14. M. Padberg and G. Rinaldi. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47:219–257, 1990.
15. M. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.
16. G. Reinelt. *The traveling salesman - Computational solutions for TSP applications*. Number 840 in Lecture Notes in Computer Science. Springer-Verlag, 1994.

A Strongly Polynomial Cut Canceling Algorithm for the Submodular Flow Problem

Satoru Iwata^{1*}, S. Thomas McCormick^{2**}, and Maiko Shigeno^{3***}

¹ Graduate School of Engineering Sciences
Osaka University, Osaka 560, Japan
`iwata@sys.es.osaka-u.ac.jp`

² Faculty of Commerce and Business Administration
University of British Columbia, BC V6T 1Z2, Canada
`stmv@adk.commerce.ubc.ca`

³ Institute of Policy and Planning Sciences
University of Tsukuba, Ibaraki 305, Japan
`maiko@shako.sk.tsukuba.ac.jp`

Abstract. This paper presents a new strongly polynomial cut canceling algorithm for minimum cost submodular flow. The algorithm is a generalization of our similar cut canceling algorithm for ordinary min-cost flow. The advantage of cut canceling over cycle canceling is that cut canceling seems to generalize to other problems more readily than cycle canceling. The algorithm scales a relaxed optimality parameter, and creates a second, inner relaxation that is a kind of submodular max flow problem. The outer relaxation uses a novel technique for relaxing the submodular constraints that allows our previous proof techniques to work. The algorithm uses the min cuts from the max flow subproblem as the relaxed most positive cuts it chooses to cancel. We show that this algorithm needs to cancel only $O(n^3)$ cuts per scaling phase, where n is the number of nodes. Furthermore, we also show how to slightly modify this algorithm to get a strongly polynomial running time.

1 Introduction

A fundamental problem in combinatorial optimization is min-cost network flow (MCF). It can be modeled as a linear program with guaranteed integral optimal solutions (with integral data), and many polynomial and strongly polynomial algorithms for it exist (see Ahuja, Magnanti, and Orlin [1] for background). Researchers in mathematical programming have developed a series of extensions of MCF having integral optimal solutions (see Schrijver [28]).

* Research supported by a Grant-in-Aid for Scientific Research from Ministry of Education, Science, Sports and Culture of Japan.

** Research supported by an NSERC Operating Grant; part of this research was done during visits to RIMS at Kyoto University and SORIE at Cornell University.

*** Research supported by a Grant-in-Aid for Scientific Research from Ministry of Education, Science, Sports and Culture of Japan.

We have long been interested in finding a *generic* (strongly) polynomial algorithm for such problems, i.e., one that is easily extended from the MCF case to more general cases. A natural class of generic algorithm to consider is the class of (primal) cycle canceling algorithms, and (dual) cut canceling algorithms. These are natural since they take improving steps coming from the linear algebra of the constraints, and whose cost depends only on the local effect on the objective value. It is (in theory) easy to figure out what these objects should look like for more general models than MCF.

A natural first step in applying such a research program is to consider the case of *submodular flow* (SF). This problem very much resembles ordinary MCF, except that the usual conservation constraints have been relaxed into constraints on the total violation of conservation on any node subset. SF was shown to enjoy integral optimal solutions by Edmonds and Giles [4]. Early algorithmic contributions came from [3,6,8]. Our algorithm uses many ideas from these papers. The first strongly polynomial algorithm for SF is due to Frank and Tardos [7], who generalized the strongly polynomial MCF algorithm by Tardos [30] to a fairly wide class of combinatorial optimization problems. A more direct generalization of the Tardos algorithm was presented by Fujishige, Röck, and Zimmermann [12] with the aid of the tree-projection method by Fujishige [9].

Unfortunately it has been surprisingly difficult to extend known MCF cycle and cut canceling algorithms to SF. One of the most attractive MCF canceling algorithms is the min mean cycle canceling algorithm of Goldberg and Tarjan [14] (and its dual, the max mean cut canceling algorithm of Ervolina and McCormick [5]). Cui and Fujishige [2] were able to show finiteness for this algorithm, but no polynomial bound. McCormick, Ervolina, and Zhou [23] show that it is unlikely that a straightforward generalization of either min mean cycle or max mean cut canceling is going to be polynomial for SF. Recently, Wallacher and Zimmermann [31] devised a weakly polynomial cycle canceling algorithm for SF based on the min ratio approach by Zimmermann [32]. This algorithm is provably not strongly polynomial.

Another possibility is to cancel cycles or cuts which maximize the violation of complementary slackness with respect to a relaxed optimality parameter, the *relaxed min/max* canceling algorithms. These algorithms allow for a simpler computation of the cycle or cut to cancel, but otherwise enjoy the relatively simple analysis of min mean cycle/max mean cut canceling. These algorithms are developed and analyzed in [29].

The same authors found a way to generalize relaxed min cycle canceling to SF [18], which led to nearly the fastest weakly polynomial running time for SF. This algorithm leads to the same strongly polynomial bound as the fastest known one by Fujishige, Röck, and Zimmermann [12], and it can also be extended to SF with separable convex costs. However, it seems unlikely that it would extend to more general cases like M-convex SF, see Murota [24,25].

This difficulty lead us to consider instead extending relaxed max cut canceling to SF. This paper will show how to modify relaxed max cut canceling in a non-

trivial way so that it will generalize to both a weakly and a strongly polynomial algorithm for SF.

The only other polynomial cut canceling algorithm we know for SF is the most helpful total cut canceling algorithm of [21]. However that algorithm is provably not strongly polynomial, and it needs an oracle to compute exchange capacities for a derived submodular function which appears to be achievable only through the ellipsoid algorithm, even if we have a non-ellipsoid oracle for the original submodular function. By contrast, the present algorithm can compute exchange capacities in derived networks using only an oracle for the original function, and is the first dual strongly polynomial algorithm for SF that we know of. Our algorithm also appears to be the first strongly polynomial algorithm (primal or dual) for SF that avoids scaling or rounding the data. Furthermore, it seems very likely to us that the present algorithm will be able to be further generalized so as to apply to the M-convex case unlike its cycle canceling cousin.

2 Submodular Flow

An instance of submodular flow looks much like an instance of MCF. We are given a directed graph $G = (N, A)$ with node set N of cardinality n and arc set A of cardinality m . We are also given *lower* and *upper bounds* $\ell, u \in \mathbf{R}^A$ on the arcs, and *costs* $c \in \mathbf{R}^A$ on the arcs.

We need some notation to talk about relaxed conservation. If $w \in \mathbf{R}^X$ and $Y \subseteq X$, then we abbreviate $\sum_{y \in Y} w_y$ by $w(Y)$ as usual. For node subset S , define $\Delta^+ S$ as $\{i \rightarrow j \in A \mid i \in S, j \notin S\}$, $\Delta^- S$ as $\{i \rightarrow j \in A \mid i \notin S, j \in S\}$, and $\Delta S = \Delta^+ S \cup \Delta^- S$. We say that arc a *crosses* S if $a \in \Delta S$. If φ is a flow on the arcs (i.e., $\varphi \in \mathbf{R}^A$), the *boundary* of φ , $\partial\varphi(S) \equiv \varphi(\Delta^+ S) - \varphi(\Delta^- S)$, is the net flow out of set S . Note that boundary is *modular*, i.e., $\partial\varphi(S) = \sum_{i \in S} \partial\varphi(\{i\})$. In the sequel we will consider several auxiliary networks whose arc sets are supersets of A , so we will often subscript ∂ and Δ by the set of arcs we want them to include at that point.

Usual MCF conservation requires that $\partial\varphi(\{i\}) = 0$ for all $i \in N$, which implies that $\partial\varphi(S) = 0$ for all $S \subseteq N$. From this perspective it is natural to relax this constraint to $\partial\varphi(S) \leq f(S)$ for some set function $f : 2^N \rightarrow \mathbf{R}$. A way to get integral optimal solutions with such a constraint is to require that f be *submodular*, e.g., that it satisfy

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

for all $S, T \subseteq N$. For a submodular function f with $f(\emptyset) = 0$, the *base polyhedron* $B(f)$ is defined by

$$B(f) = \{x \mid x \in \mathbf{R}^N, x(N) = f(N), \forall S \subseteq N : x(S) \leq f(S)\}.$$

A vector in $B(f)$ is called a *base*. Since $\partial\varphi(\emptyset) = \partial\varphi(N) = 0$ for any flow φ , requiring that $f(N) = 0$ and $\partial\varphi(S) \leq f(S)$ for all $S \subseteq N$ is equivalent to requiring that $\partial\varphi$ belong to $B(f)$.

Now if $f : 2^N \rightarrow \mathbf{R}$ is a submodular function with $f(\emptyset) = f(N) = 0$, then the submodular flow problem is:

$$\begin{aligned} & \text{Minimize } \sum_{a \in A} c_a \varphi_a \\ & \text{subject to } \ell_a \leq \varphi_a \leq u_a \quad (a \in A) \\ & \quad \partial \varphi \in B(f). \end{aligned} \tag{1}$$

This is a linear program with an exponential number of constraints.

Technically, the dual problem to (1) should have an exponential number of dual variables, one for each subset of N . However it is possible ([10]) and much more convenient to reduce these down to dual variables on nodes, or *node potentials* $\pi \in \mathbf{R}^N$. For arc $a = i \rightarrow j$, we define the *reduced cost* on a by $c_a^\pi = c_a + \pi_i - \pi_j$.

We need some further concepts to characterize optimality for (1). If φ is a flow with boundary $x = \partial \varphi$ such that $x \in B(f)$, then we say that subset S is φ -tight, or x -tight, or just *tight* if $x(S) (= \partial \varphi(S)) = f(S)$. Submodularity implies that the union and intersection of tight sets are tight.

If π is a set of node potentials with distinct values $v_1 > v_2 > \dots > v_h$, then the k th *level set* of π is $L_k^\pi \equiv \{i \in N \mid \pi_i \geq v_k\}$, $k = 1, 2, \dots, h$. It will be convenient to let $L_0^\pi = \emptyset$. Note that $\emptyset = L_0^\pi \subset L_1^\pi \subset L_2^\pi \subset \dots \subset L_h^\pi = N$.

The “submodular part” of the optimality condition requires that an optimal φ and π satisfy that each L_k^π is φ -tight. There are two equivalent ways to express this. The first is to require that $\partial \varphi$ maximize the objective $\pi^T x$ over $x \in B(f)$. The second requires defining

$$f^\pi(S) = \sum_{k=1}^h \{f((S \cap L_k^\pi) \cup L_{k-1}^\pi) - f(L_{k-1}^\pi)\} \quad (S \subseteq N).$$

We note that f^π is submodular, and that $f^\pi \leq f$. It is easy to see that if S nests with each L_k^π (either $S \subseteq L_k^\pi$ or $L_k^\pi \subseteq S$), then $f^\pi(S) = f(S)$. Also, x is in $B(f^\pi)$ if and only if $x \in B(f)$ and every L_k^π is x -tight for f [3]. Thus the second equivalence is to require that $\partial \varphi \in B(f^\pi)$.

Theorem 1 ([10, Theorem 5.3]). *A submodular flow φ is optimal if and only if there exists a node potential $\pi : N \rightarrow \mathbf{R}$ such that*

$$\begin{aligned} c_a^\pi > 0 &\Rightarrow \varphi_a = \ell_a, \\ c_a^\pi < 0 &\Rightarrow \varphi_a = u_a, \end{aligned}$$

and $\partial \varphi \in B(f^\pi)$.

3 Dual Approximate Optimality

We first cover the details of how to check node potentials π for optimality. We then show how to adapt checking exact optimality to checking a new kind of

approximate optimality. This checking routine will form the core of our cut canceling algorithm, as it will produce the cuts that we cancel. For this paper a *cut* is just a subset S of N . We cancel a cut by increasing π_i by some step length β for $i \in S$, which has the effect of increasing c_a^π on $\Delta^+ S$, decreasing c_a^π on $\Delta^- S$, and changing the level sets of π .

Given a node potential π , we define *modified bounds* ℓ^π and u^π as follows. If $c_a^\pi < 0$ then $\ell_a^\pi = u_a^\pi = \ell_a$; if $c_a^\pi = 0$ then $\ell_a^\pi = \ell_a$ and $u_a^\pi = u_a$; if $c_a^\pi > 0$ then $\ell_a^\pi = u_a^\pi = u_a$. Then Theorem 1 implies that π is optimal if and only if there is a feasible flow in the network G^π with bounds ℓ^π , u^π , and f^π .

For the proof of correctness of our algorithm we will need some details of how feasibility of G^π is checked.

We use a variant of an algorithm of Frank [6] to check feasibility of G^π . It starts with any initial base $x \in B(f^\pi)$ and any initial φ satisfying ℓ^π and u^π . We now define a *residual network* on N . If $\varphi_{ij} > \ell_{ij}^\pi$, then we have a backward residual arc $j \rightarrow i$ with residual capacity $r_{ij} = \varphi_{ij} - \ell_{ij}^\pi$; if $\varphi_{ij} < u_{ij}^\pi$, then we have a forward residual arc $i \rightarrow j$ with $r_{ji} = u_{ij}^\pi - \varphi_{ij}$. To deal with relaxed conservation, we have *jumping arcs*: For every $i, j \in N$ with $i \neq j$ and $x \in B(f^\pi)$, define

$$\tilde{r}(x, i, j) = \max\{\alpha \mid x + \alpha\chi_i - \alpha\chi_j \in B(f^\pi)\};$$

thus $\tilde{r}(x, i, j) > 0$ means that there is no x -tight set containing i but not j . Make a jumping arc $j \rightarrow i$ with capacity $\tilde{r}(x, i, j)$ whenever $\tilde{r}(x, i, j) > 0$. Note that S is x -tight if and only if there are no jumping arcs w.r.t. x entering S .

The Feasibility Algorithm finds directed residual paths from $N^+ = \{i \in N \mid x_i > \partial\varphi(\{i\})\}$ to $N^- = \{i \in N \mid x_i < \partial\varphi(\{i\})\}$ with a minimum number of arcs in the residual network. On each path it augments flow φ on the residual arcs, and modifies x as per the jumping arcs, which monotonically reduces the difference of x and $\partial\varphi$. By using a lexicographic selection rule, the algorithm terminates in finite time. At termination, either x coincides with $\partial\varphi$, which implies the δ -optimality of π ; or there is no directed path from N^+ to N^- .

In this last case, define $T \subseteq N$ as the set of nodes from which N^- is reachable by directed residual paths. No jumping arcs enter T , so it must be tight for the final x , and it must contain all i with $x_i < \partial\varphi(\{i\})$. Furthermore, we have $\varphi(\Delta^+ T) = \ell^\pi(\Delta^+ T)$ and $\varphi(\Delta^- T) = u^\pi(\Delta^- T)$. Thus we get

$$V^\pi(T) \equiv \ell^\pi(\Delta_A^+ T) - u^\pi(\Delta_A^- T) - f^\pi(T) = \partial\varphi(T) - x(T) > 0.$$

We call a node subset S with $V^\pi(S) > 0$ a *positive cut*. Similar reasoning shows that for any other $S \subseteq N$, we have $V^\pi(S) \leq \partial\varphi(S) - x(S) \leq \partial\varphi(T) - x(T) = V^\pi(T)$, proving that T is a *most positive cut*. Intuitively, $V^\pi(T)$ measures how far away from optimality π is. We summarize as follows:

Lemma 1. *Node potentials π are optimal if and only if there are no positive cuts w.r.t. π . When π is not optimal, the output of the Feasibility Algorithm is a most positive cut.*

We denote the running time of the Feasibility Algorithm by FA. As usual, we assume that we have an oracle to compute exchange capacities, and denote its

running time by h . Computing an exchange capacity is a submodular function minimization on a distributive lattice, which can be done via the ellipsoid method in (strongly) polynomial time [15]. Cunningham and Frank [3, Theorem 7] show how to derive an oracle for computing exchange capacities for f^π from an oracle for f with the same running time. Fujishige and Zhang [11] show how to extend the push-relabel algorithm of Goldberg and Tarjan [13] to get $\text{FA} = O(n^3 h)$.

Unfortunately, even for min-cost flow, an example of Hassin [16] shows that it may be necessary to cancel an exponential number of most positive cuts to achieve optimality. In [29] we get around this difficulty for MCF by relaxing the optimality conditions by a parameter $\delta > 0$, and then applying scaling to δ . It then turns out that only a polynomial number of relaxed most positive cuts need to be canceled for a given value of the parameter, and only a polynomial number of scaled subproblems need to be solved.

Until now it has been difficult to find a relaxation in the submodular flow case that would allow the same analysis to apply. Here we introduce a new relaxation that works. We relax the modified bounds on the arcs in A by widening them by δ just as in [5]. Define $\ell_a^{\pi, \delta} = \ell_a^\pi - \delta$ and $u_a^{\pi, \delta} = u_a^\pi + \delta$ for every arc $a \in A$.

We also need to relax the submodular bounds on conservation by some function of δ . Define $f^{\pi, \delta}(S) = f^\pi(S) + \delta|S| \cdot |N - S|$. This relaxation is closely related to [17]. Since $|S| \cdot |N - S|$ is submodular, $f^{\pi, \delta}$ is submodular.

We can now define the relaxed cut value of S as

$$V^{\pi, \delta}(S) = \ell^{\pi, \delta}(\Delta_A^+ S) - u^{\pi, \delta}(\Delta_A^- S) - f^{\pi, \delta}(S).$$

We say that node potential π is δ -optimal if $V^{\pi, \delta}(S) \leq 0$ holds for every cut S . Thus π is 0-optimal if and only if it is optimal.

To check if π is δ -optimal, we need to see if there is a feasible flow in the network with bounds $\ell^{\pi, \delta}$, $u^{\pi, \delta}$, and $f^{\pi, \delta}$. It turns out to be more convenient to move the δ relaxation off $f^{\pi, \delta}$ onto a new set of arcs. Define $\widehat{G}^\pi = (N, A \cup E)$ to be the directed graph obtained from G by adding $E = \{i \rightarrow j \mid i, j \in N, i \neq j\}$ to the arc set. Extend the bounds ℓ^π , u^π , $\ell^{\pi, \delta}$ and $u^{\pi, \delta}$ from A to E by setting $\ell_e^\pi = u_e^\pi = 0$, $\ell_e^{\pi, \delta} = 0$ and $u_e^{\pi, \delta} = \delta$ for every $e \in E$. For convenience set $I = A \cup E$ and $m' = |A \cup E| = m + n(n-1)$.

Now checking $\ell^{\pi, \delta}$, $u^{\pi, \delta}$ and $f^{\pi, \delta}$ for feasibility on G (with arc set A) is equivalent to checking $\ell^{\pi, \delta}$, $u^{\pi, \delta}$, and f^π for feasibility on $\widehat{G}^\pi$ (with arc set $I = A \cup E$). Also, the relaxed cut value on G can be re-expressed as

$$V^{\pi, \delta}(S) = \ell^{\pi, \delta}(\Delta_I^+ S) - u^{\pi, \delta}(\Delta_I^- S) - f^\pi(S).$$

If there is a feasible submodular flow in $\widehat{G}^\pi$, then π is δ -optimal. Otherwise, the Feasibility Algorithm gives us a node subset T that maximizes $V^{\pi, \delta}(S)$, a *relaxed most positive cut* or δ -MPC. When π is not δ -optimal, the Feasibility Algorithm also gives us an optimal “max flow” φ on $\widehat{G}^\pi$ and a base x in $B(f^\pi)$ such that $x_i \leq \partial_I \varphi(\{i\})$ for $i \in T$ and $x_i \geq \partial_I \varphi(\{i\})$ for $i \notin T$.

For Section 5 we will need to consider max mean cuts. The *mean value* of node subset S is

$$\overline{V}^\pi(S) \equiv \frac{V^\pi(S)}{|\Delta_A S| + |S| \cdot |N - S|}.$$

A *max mean cut* attains the maximum in $\delta(\pi) \equiv \max_S \bar{V}^\pi(S)$. By standard LP duality arguments $\delta(\pi)$ also equals the minimum δ such that there is a feasible flow in $\hat{G}^\pi$ with bounds $\ell^{\pi,\delta}$, $u^{\pi,\delta}$, and f^π . Define U to be the maximum absolute value of any ℓ_a , u_a , or $f(\{i\})$. A max mean cut can be computed using $O(\min\{m', \frac{\log(nU)}{1+\log \log(nU)-\log \log n}\})$ calls to the Feasibility Algorithm in the framework of Newton's Algorithm, see Ervolina and McCormick [22] or Radzik [26].

4 Cut Cancellation

We will start out with δ large and drive δ towards zero, since π is 0-optimal if and only if it is optimal. The next lemma says that δ need not start out too big, and need not end up too small. Its proof is similar to [5, Lemma 5.1].

Lemma 2. *Suppose that ℓ , u , and f are integral. Then any node potentials π are $2U$ -optimal. Moreover, when $\delta < 1/m'$, any δ -optimal node potentials are optimal.*

Our relaxed δ -MPC Canceling algorithm will start with $\delta = 2U$ and will execute scaling phases, where each phase first sets $\delta := \delta/2$. The input to a phase will be a 2δ -optimal set of node potentials from the previous phase, and its output will be a δ -optimal set of node potentials. Lemma 2 says that after $O(\log(nU))$ scaling phases we have $\delta < 1/m'$, and we are optimal. Within a scaling phase we use the Feasibility Algorithm to find a δ -MPC T . We then *cancel* T by adding a constant *step length* β to π_i for each node in T to get π' .

In ordinary min-cost flow we choose the step length β based on the first arc in A whose reduced cost hits zero (as long as the associated flow is within the bounds, see [29, Figure 2]). This bound on β is

$$\eta \equiv \min \left\{ \begin{array}{l} \min\{c_a^\pi \mid a \in \Delta_A^+ T, c_a^\pi < 0, \varphi_a \geq \ell_a\} \\ \min\{c_a^\pi \mid a \in \Delta_A^- T, c_a^\pi > 0, \varphi_a \leq u_a\} \end{array} \right\}.$$

(Note that optimality of T implies that every arc a with $\varphi_a \geq \ell_a$ has negative reduced cost, and every arc a with $\varphi_a \leq u_a$ has positive reduced cost.) We say that an arc achieving the min for η *determines* η .

Here we must also worry about the level set structure of π changing during the cancel. We need to increase flows on jumping arcs leaving T so that the L_k^π will be tight. We try to do this by decreasing flow on arcs of $\Delta_E^- T$, all of which have flow δ since T is tight. If the exchange capacity of such an arc is at most δ we can do this. Otherwise, this E -arc will determine β , and the large exchange capacity will allow us to show that the algorithm makes sufficient progress.

These considerations lead to the subroutine ADJUSTFLOW below. It takes as input the optimal max flow φ from the Feasibility Algorithm, and its associated base $x \in B(f^\pi)$. It computes an update $\psi \in \mathbf{R}^E$ to φ and base x' so that x' will be the base associated with $\varphi' \equiv \varphi - \psi$. Note that in ADJUSTFLOW we always

Algorithm ADJUSTFLOW:**begin** $\psi_e = 0$ **for** $e \in E$; $H := \{j \rightarrow i \mid i \in T, j \in N - T, 0 < \pi_j - \pi_i < \eta\}$;**while** $H \neq \emptyset$ **do****begin** $x' := x - \partial_E \psi$;select $e = j \rightarrow i \in H$ with minimum $\pi_j - \pi_i$;set $H := H - \{e\}$;**if** $\tilde{r}(x', i, j) < \delta$ **then** $\psi_e := \tilde{r}(x', i, j)$;**else return** $\beta := \pi_j - \pi_i$ [β is determined by jumping arc $j \rightarrow i$];**end****return** $\beta := \eta$ [β is determined by the A -arc determining η];**end.**

compute $\tilde{r}(x, i, j)$ w.r.t. f , never w.r.t. f^π , so that jumping arcs are w.r.t. f , not f^π . We call an arc that determines β a *determining arc*.

The full description of canceling a cut is now: Use the Feasibility Algorithm to find δ -MPC T , max flow φ , and base x . Run ADJUSTFLOW to modify φ to $\varphi' = \varphi - \psi$ and x to $x' = x - \partial_E \psi$, and to choose β . Finally, increase π_i by β for all $i \in T$. A scaling phase cancels δ -MPCs in this manner until π is δ -optimal. The next lemma shows that canceling a δ -MPC cannot increase the δ -MPC value, and that $V^{\pi, \delta}(S)$ will decrease by at least δ under some circumstances.

Lemma 3. *Suppose we cancel a δ -MPC T for π to get a node potential π' . Then $V^{\pi', \delta}(S) \leq V^{\pi, \delta}(T)$ holds for any cut S .*

Proof. It is easy to show similarly to [29, Lemma 5.3] that $\ell_a^{\pi', \delta} \leq \varphi_a \leq u_a^{\pi', \delta}$ holds for every $a \in A$. As a result of ADJUSTFLOW, we obtain a flow $\psi \in \mathbf{R}^E$ with $0 \leq \psi_e < \delta$ for $e \in E$. Put $\varphi'_a = \varphi_a$ for $a \in A$ and $\varphi'_e = \varphi_e - \psi_e$ for $e \in E$. To prove that φ' is feasible for $B(f^{\pi'})$ we need:

Claim: $x' = x - \partial_E \psi \in B(f^{\pi'})$.

Proof: If the initial $H = \emptyset$ in ADJUSTFLOW (i.e., $\pi_j > \pi_i$ implies that $\pi_j \geq \pi_i + \eta$), then $\psi \equiv 0$ and so $x' = x$. In this case it can be shown that $f^{\pi'}(S) = f^\pi(S \cap T) + f^\pi(S \cup T) - f^\pi(T)$. Since $x(T) = x'(T) = f^\pi(T)$, we have $x'(S) = x'(S \cap T) + x'(S \cup T) - x'(T) \leq f^\pi(S \cap T) + f^\pi(S \cup T) - f^\pi(T) = f^{\pi'}(S)$.

Suppose instead that the initial H is non-empty. Here we know at least that $x' \in B(f)$ (since ψ_{ji} is never bigger than $\tilde{r}(x', i, j)$). Denote $T \cap (L_k^\pi - L_{k-1}^\pi)$ by T_k , and $(L_k^\pi - L_{k-1}^\pi) - T$ by $\overline{T}_k$, so that the T_k partition T , and the $\overline{T}_k$ partition $N - T$. Then a typical level set of π' looks like $L' = (\bigcup_{k=1}^q T_k) \cup (\bigcup_{k=1}^p \overline{T}_k) = L_p^\pi \cup (T \cap L_q^\pi)$ for some $0 \leq p < q \leq h$.

Now both T and L_i^π are φ -tight for f^π , so their union and intersection are both φ -tight for f^π . By the same reasoning, $\widehat{T}_i \equiv (T \cap L_i^\pi) \cup L_{i-1}^\pi = (\bigcup_{k=1}^i T_k) \cup (\bigcup_{k=1}^{i-1} \overline{T}_k)$ is also φ -tight for f^π . Since every arc of H starts in some $\overline{T}_j$ and ends in some T_i with $j < i$, no arc of H crosses any $\widehat{T}_i$, so each $\widehat{T}_i$ is also φ' -tight for

f^π . Each $\widehat{T}_i$ nests with the L_k^π , so we have $f^\pi(\widehat{T}_i) = f(\widehat{T}_i)$, so each $\widehat{T}_i$ is in fact φ' -tight for f . This implies that the only possible jumping arcs entering level set L' of π' are those from $\overline{T}_j$ to T_i for $p < j < i \leq q$. But these jumping arcs all belong to H and were removed by ADJUSTFLOW. Thus L' is φ' -tight for f , and so $x' \in B(f^{\pi'})$. $\square$

Since $\psi_e > 0$ implies $\varphi_e = \delta$, every $e \in E$ satisfies $0 \leq \varphi'_e \leq \delta$. Therefore we have

$$\begin{aligned}
 V^{\pi',\delta}(S) &= \ell^{\pi',\delta}(\Delta_I^+ S) - u^{\pi',\delta}(\Delta_I^- S) - f^{\pi'}(S) \quad (\text{def'n of } V^{\pi',\delta}(S)) \\
 &\leq \partial_I \varphi'(S) - x'(S) \quad (\text{feas. of } \varphi', x' \in B(f^{\pi'})) \\
 &= \partial_I \varphi(S) - x(S) \quad (\text{def'n of } \varphi', x') \\
 &\leq \partial_I \varphi(T) - x(T) \quad (T \text{ a } \delta\text{-MPC}) \\
 &= \ell^{\pi,\delta}(\Delta_I^+ T) - u^{\pi,\delta}(\Delta_I^- T) - f^\pi(T) \quad (T \text{ is } \varphi\text{-tight for } f^\pi) \\
 &= V^{\pi,\delta}(T). \quad (\text{def'n of } V^{\pi,\delta}(T))
 \end{aligned}$$

$\square$

Corollary 1. *Suppose we cancel a δ -MPC T for π to get a node potential π' . If the determining arc $a \in A \cup E$ crosses S , then we have $V^{\pi',\delta}(S) \leq V^{\pi,\delta}(T) - \delta$.*

Proof. If the determining arc $a \in A$, then $\ell_a^{\pi',\delta} + \delta = \ell_a \leq \varphi_a \leq u_a = u_a^{\pi',\delta} - \delta$, and hence $\ell^{\pi',\delta}(\Delta_I^+ S) - u^{\pi',\delta}(\Delta_I^- S) \leq \partial_I \varphi'(S) - \delta$. If $a \in \Delta_E^+ S$, then $\varphi_a = \delta = \ell_a^{\pi',\delta} + \delta$, and hence $\ell^{\pi',\delta}(\Delta_I^+ S) - u^{\pi',\delta}(\Delta_I^- S) \leq \partial_I \varphi'(S) - \delta$. If $a \in \Delta_E^- S$, then $f(S) - x'(S) \geq \delta$, and hence $f^{\pi'}(S) \geq x'(S) + \delta$. In either case we have $V^{\pi',\delta}(S) \leq \partial_I \varphi'(S) - x'(S) - \delta$, which implies $V^{\pi',\delta}(S) \leq V^{\pi,\delta}(T) - \delta$. $\square$

Note that we have two different notions of “closeness to optimality” in this algorithm. At the outer level of scaling phases we drive δ towards zero (and so π towards optimality), and inside a phase we drive the δ -MPC value towards zero (and so π towards δ -optimality). Observe that the difference between $\partial\varphi$ and x in the Feasibility Algorithm is an inner relaxation of the condition that π is δ -optimal, in that we keep a flow satisfying the bounds $\ell^{\pi,\delta}$ and $u^{\pi,\delta}$ and a base x in $B(f^\pi)$. As the phase progresses the difference is driven to zero, until the final flow proves δ -optimality of the final π .

We can now prove the key lemma in the convergence proof of δ -MPC Canceling, which shows that we must be in a position to apply Lemma 3 reasonably often.

Lemma 4. *After at most n cut cancellations, the value of a δ -MPC decreases by at least δ .*

Proof. Suppose that the first cancellation has initial node potentials π^0 and cancels δ -MPC T^1 to get π^1 , the next cancels T^2 to get π^2 , $\dots$, and the n th cancels T^n to get π^n . Each cancellation makes at least one arc into a determining arc. Consider the subgraph of these determining arcs. If a cut creates a determining arc whose ends are in the same connected component of this subgraph, then this cut must be crossed by a determining arc from an earlier cut. We can avoid

this only if each new determining arc strictly decreases the number of connected components in the subgraph. This can happen at most $n - 1$ times, so it must happen at least once within n iterations.

Let k be an iteration where T^k shares a determining arc with an earlier cut T^h . By Corollary 1 applied to $T = T^h$ and $S = T^k$, we have $V^{\pi^{h-1}, \delta}(T^h) \geq V^{\pi^h, \delta}(T^k) + \delta$. Note that the δ -MPC value at iteration i is $V^{\pi^{i-1}, \delta}(T^i)$. If $V^{\pi^h, \delta}(T^k) \geq V^{\pi^{k-1}, \delta}(T^k)$, Lemma 3 says that $V^{\pi^0, \delta}(T^1) \geq V^{\pi^{h-1}, \delta}(T^h) \geq V^{\pi^h, \delta}(T^k) + \delta \geq V^{\pi^{k-1}, \delta}(T^k) + \delta \geq V^{\pi^{n-1}, \delta}(T^n) + \delta$.

If instead $V^{\pi^h, \delta}(T^k) < V^{\pi^{k-1}, \delta}(T^k)$, then let p be the latest iteration between h and k with $V^{\pi^{p-1}, \delta}(T^k) < V^{\pi^p, \delta}(T^k)$. The only way for the value of T^k to increase like this is if there is an arc $a \in A$ with $c_a^{\pi^{p-1}} = 0$ that crosses T^k in the reverse orientation to its orientation in T^p , or $f_a^{\pi^{p-1}, \delta}(T^k) > f_a^{\pi^p, \delta}(T^k)$ holds. Let φ^p be a flow proving optimality of T^p . If $a \in \Delta_A^+ T^p \cap \Delta_A^- T^k$ ($a \in \Delta_A^- T^p \cap \Delta_A^+ T^k$) then we have $u_a^{\pi^p} = \varphi_a^p + 2\delta \geq \varphi_a^p + \delta$ ($l_a^{\pi^p} = \varphi_a^p - 2\delta \leq \varphi_a^p - \delta$). When $f_a^{\pi^{p-1}, \delta}(T^k) > f_a^{\pi^p, \delta}(T^k)$ holds, there exist $i, j \in N$ such that $i \in T^k \setminus T^p$, $j \in T^p \setminus T^k$. It follows from $i \rightarrow j \in \Delta_I^- T^p$ and $j \rightarrow i \in \Delta_I^+ T^p$ that $\varphi_e^p = \delta = l_e^{\pi^p, \delta} + \delta$ for $e = i \rightarrow j$ and $\varphi_{e'}^p = 0 = u_{e'}^{\pi^p, \delta} - \delta$ for $e' = j \rightarrow i$. In either case, we have $l^{\pi^p, \delta}(\Delta_I^+ T^k) - u^{\pi^p, \delta}(\Delta_I^- T^k) \leq \partial_I \varphi^p - \delta$. Thus equations of the proof of Lemma 3 applies, showing that $V^{\pi^{p-1}, \delta}(T^p) - \delta \geq V^{\pi^p, \delta}(T^k)$. Then Lemma 3 and the choice of p says that $V^{\pi^0, \delta}(T^1) \geq V^{\pi^{p-1}, \delta}(T^p) \geq V^{\pi^p, \delta}(T^k) + \delta \geq V^{\pi^{k-1}, \delta}(T^k) + \delta \geq V^{\pi^{n-1}, \delta}(T^n) + \delta$. $\square$

Lemma 5. *The value of every δ -MPC in a scaling phase is at most $m'\delta$.*

Proof. Let φ prove the 2δ -optimality of the initial π in a phase, so that φ violates the bounds l^π and u^π by at most 2δ on every arc of $A \cup E$, and $\varphi_{si} = 0$ for all i . To get an initial flow φ^0 to begin the next phase that violates the bounds by at most δ , we need to change φ by at most δ on each arc of $A \cup E$. Then if we set $\varphi_{si}^0 = \delta |\Delta_I^+ \{i\}|$ for all i , φ^0 will satisfy f^π also. The sum $\sum_{i \in N} \varphi_{si}^0$ is at most $m'\delta$, and this is an upper bound on the value of the first δ -MPC in this phase. By Lemma 3, it is then also a bound on the value of every δ -MPC in the phase. $\square$

Putting Lemmas 4 and 5 together yields our first bound on the running time of δ -MPC Canceling:

Theorem 2. *A scaling phase of δ -MPC Canceling cancels at most $m'n$ δ -MPCs. Thus the running time of δ -MPC Canceling is $O(n^3 \log(nU) \text{FA}) = O(n^6 h \log(nU))$.*

Proof. Lemma 5 shows that the δ -MPC value of the first cut in a phase is at most $m'\delta$. It takes at most n iterations to reduce this by δ , so there are at most $m'n = O(n^3)$ iterations per phase. The time per iteration is dominated by computing a δ -MPC, which is $O(\text{FA})$. The number of phases is $O(\log(nU))$ by Lemma 2. $\square$

5 A Strongly Polynomial Bound

We first prove the following lemma. Such a result was first proved by Tardos [30] for MCF, and this version is an extension of dual versions by Fujishige [9] and Ervolina and McCormick [5].

Lemma 6. *Suppose that flow φ' proves that π' is δ' -optimal. If arc $a \in A$ satisfies $\varphi'_a < u_a - (m' + 1)\delta'$, then all optimal π^* have $c_a^{\pi^*} \geq 0$. If arc $a \in A$ satisfies $\varphi'_a > \ell_a + (m' + 1)\delta'$, then all optimal π^* have $c_a^{\pi^*} \leq 0$. If $\partial_I \varphi'(T^0) > f^{\pi^0}(T^0) + (m'n/2)\delta'$ for some π^0 and some $T^0 \subseteq N$, then there is an E -arc $i \rightarrow j$ leaving some level set of π^0 such that all optimal π^* have $\pi_i^* \leq \pi_j^*$.*

Proof. Note that φ' proving δ' -optimality of π' implies that there is no jumping arc entering any $L_k^{\pi'}$ for any k . Since $x' \equiv \partial_I \varphi' \in B(f^{\pi'})$, this says that x' is a π' -maximum base of $B(f)$.

Now change φ' , a flow on $A \cup E$ feasible for $\ell^{\pi', \delta'}$ and $u^{\pi', \delta'}$, into flow $\widehat{\varphi}$ on just A , feasible for $\ell^{\pi'}$ and $u^{\pi'}$, by getting rid of φ'_e for $e \in E$, and by changing φ'_a by at most δ' on each $a \in A$. Note that this $\widehat{\varphi}$ will not in general satisfy $\partial_A \widehat{\varphi} \in B(f)$, but that $\widehat{\varphi}$ otherwise satisfies all optimality conditions: it satisfies the bounds and is complementary slack with π' . Define $N^+ = \{i \in N \mid x'_i > \partial_A \widehat{\varphi}(\{i\})\}$ and $N^- = \{i \in N \mid x'_i < \partial_A \widehat{\varphi}(\{i\})\}$, so that

$$x'(N^+) \leq m'\delta' + \partial_A \widehat{\varphi}(N^+). \quad (2)$$

We now apply the successive shortest path algorithm for submodular flow of Fujishige [8] starting from $\widehat{\varphi}$. (As originally stated, this algorithm is finite only for integer data, but the lexicographic techniques of Schönsleben [27] and Lawler and Martel [20] show that it can be made finite for any data.) This algorithm looks for an augmenting path from a node $i \in N^+$ to a node $j \in N^-$, where residual capacities on A -arcs come from $\widehat{\varphi}$, and residual capacities on jumping arcs come from x' . It chooses a shortest augmenting path (using the current reduced costs as lengths; such a path can be shown to always exist) and augments flow along the path, updating π' by the shortest path distances, and x' as per the jumping arcs. This update maintains the properties that the current $\widehat{\varphi}$ satisfies the bounds and is complementary slack with the current π' , and the current x' belongs to $B(f)$. The algorithm terminates with optimal flow φ^* once the boundary of the current $\widehat{\varphi}$ coincides with the current x' . By (2), the total amount of flow pushed by this algorithm is at most $m'\delta'$.

This implies that for each $a \in A$, $\widehat{\varphi}_a$ differs from φ_a^* by at most $m'\delta'$, so φ'_a differs from φ_a^* by at most $(m' + 1)\delta'$. In particular, if $\varphi'_a < u_a - (m' + 1)\delta'$, then $\varphi_a^* < u_a$, implying that $c_a^{\pi^*} \geq 0$, and similarly for $\varphi'_a > \ell_a + (m' + 1)\delta'$.

Suppose that P is an augmenting path chosen by the algorithm, and that flow is augmented by amount τ_P along P . Since P has at most $n/2$ jumping arcs, the boundary of any $S \subseteq N$ changes by at most $(n/2)\tau_P$ due to P . Since $\sum_P \tau_P \leq m'\delta'$ by (2), the total change in $\partial_A \widehat{\varphi}(S)$ during the algorithm is at most $(m'n/2)\delta'$. Since $|\partial_I \varphi'(S) - \partial_A \widehat{\varphi}(S)| \leq m'\delta'$, the total change in $\partial_I \varphi'(S)$ is at most $(m'n/2)\delta'$. Thus $\partial_I \varphi'(T^0) > f^{\pi^0}(T^0) + (m'n/2)\delta'$ implies that $\partial_A \varphi^*(T^0) >$

$f^{\pi^0}(T^0)$. This says that some level set of π^0 is not φ^* -tight. This implies that there is some E -arc $i \rightarrow j$ with $\pi_i^0 > \pi_j^0$ but $\pi_i^* \leq \pi_j^*$. $\square$

We now modify our algorithm a little bit. We divide our scaling phases into *blocks* of $\log_2(m' + 1)$ phases each. At the beginning of each block we compute a max mean cut T^0 with mean value $\delta^0 = \delta(\pi^0)$ and cancel T^0 (including calling ADJUSTFLOW). This ensures that our current flow is δ^0 -optimal, so we set $\delta = \delta^0$ and start the block of scaling phases. It will turn out that only $2m'$ blocks are sufficient to attain optimality.

Theorem 3. *This modified version of the algorithm takes $O(n^5 \log n \text{FA}) = O(n^8 h \log n)$ time.*

Proof. Let T^0 be the max mean cut canceled at the beginning of the block, with associated node potential π^0 , flow φ^0 , and mean value δ^0 . Complementary slackness for T^0 implies that $\varphi_a^0 = u^{\pi^0} + \delta^0$ for all $a \in \Delta_I^- T^0$, that $\varphi_a^0 = \ell^{\pi^0} - \delta^0$ for all $a \in \Delta_A^+ T^0$, that $\varphi_a^0 = \ell^{\pi^0}$ for all $a \in \Delta_I^+ T^0$, and that $\partial_I \varphi^0(T^0) = f^{\pi^0}(T^0)$. Let π' , φ' , and δ' be the similar values after the last phase in the block. Since each scaling phase cuts δ in half, we have $\delta' < \delta^0/(m' + 1)$.

Subtracting $\varphi'(\Delta_I^+ T^0) - \varphi'(\Delta_I^- T^0) - \partial_I \varphi'(T^0) = 0$ from $V^{\pi^0}(T^0)$ yields

$$\begin{aligned} (|\Delta_A T^0| + |T^0| \cdot |N - T^0|) \delta^0 &= V^{\pi^0}(T^0) \\ &= (\ell^{\pi^0} - \varphi')(\Delta_I^+ T^0) + (\varphi' - u^{\pi^0})(\Delta_I^- T^0) \\ &\quad + (\partial_I \varphi'(T^0) - f^{\pi^0}(T^0)). \end{aligned} \quad (3)$$

Now apply Lemma 6 to φ' and π' . If the term for arc a of $(\ell^{\pi^0} - \varphi')(\Delta_I^+ T^0)$ is at least $\delta^0 > (m' + 1)\delta'$, then we must have that $\ell_a^{\pi^0} = u_a$ and so $\varphi'_a < u_a - (m' + 1)\delta'$, and we can conclude that $c_a^{\pi^*} \geq 0$. But each a in $\Delta_A^+ T^0$ had $c_a^{\pi^0} < 0$, so this is a new sign constraint on c^{π} . The case for terms of $(\varphi' - u^{\pi^0})(\Delta_I^- T^0)$ is similar.

Suppose instead that all the terms in the $\Delta_A^+ T^0$ and $\Delta_A^- T^0$ sums of (3) are at most δ^0 . The total of all the E -arc terms is at most $|T^0| \cdot |N - T^0| \delta'$. Therefore the only possibility left to achieve the large left-hand side of (3) is to have $\partial_I \varphi'(T^0) > f^{\pi^0}(T^0) + (m'n/2)\delta'$. Lemma 6 says that in this case there must be a jumping arc $i \rightarrow j$ leaving some level set of π^0 such that all optimal π^* have $\pi_i^* \leq \pi_j^*$. Since π_i^0 was larger than π_j^0 , this is a new sign restriction on π .

In either case each block imposes a new sign restriction on $c_a^{\pi^*}$ for some I -arc a . At most $2m'$ such sign restrictions can be imposed before π^* is completely determined, so after at most $2m' = O(n^2)$ blocks we must be optimal. Each block requires $\log(m' + 1) = O(\log n)$ scaling phases. The proof of Theorem 2 shows that each scaling phase costs $O(n^3 \text{FA})$ time exclusive of computing the max mean cut. The time to compute a max mean cut is $O(m' \text{FA})$, which is not a bottleneck. $\square$

6 Directions for Further Research

These techniques should lead to a (strongly) polynomial maximum mean cut canceling algorithm for the submodular flow problem. It should be very straightforward to extend this algorithm to the separable convex submodular flow problem as was done for its cycle canceling cousin. We are also optimistic about extending it to the M-convex cost submodular flow problem [24,25]. If we can do this, then we would have a unified approach to a variety of optimization problems including separable convex cost optimization in totally unimodular spaces [19].

Acknowledgment

We heartily thank Lisa Fleischer for many valuable conversations about this paper and helpful comments on previous drafts of it.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin: *Network Flows — Theory, Algorithms, and Applications*, Prentice Hall, 1993.
2. W. Cui and S. Fujishige: A primal algorithm for the submodular flow problem with minimum-mean cycle selection, *J. Oper. Res. Soc. Japan*, **31** (1988), 431–440.
3. W. H. Cunningham and A. Frank: A primal-dual algorithm for submodular flows, *Math. Oper. Res.*, **10** (1985), 251–262.
4. J. Edmonds and R. Giles: A min-max relation for submodular functions on graphs, *Ann. Discrete Math.*, **1** (1977), 185–204.
5. T. R. Ervolina and S. T. McCormick: Two strongly polynomial cut canceling algorithms for minimum cost network flow, *Discrete Appl. Math.*, **46** (1993), 133–165.
6. A. Frank: Finding feasible vectors of Edmonds-Giles polyhedra, *J. Combinatorial Theory*, **B36** (1984), 221–239.
7. A. Frank and É. Tardos: An application of simultaneous Diophantine approximation in combinatorial optimization, *Combinatorica*, **7** (1987), 49–65.
8. S. Fujishige: Algorithms for solving the independent-flow problems, *J. Oper. Res. Soc. Japan*, **21** (1978), 189–204.
9. S. Fujishige: Capacity-rounding algorithm for the minimum-cost circulation problem: A dual framework of the Tardos algorithm, *Math. Programming*, **35** (1986), 298–309.
10. S. Fujishige: *Submodular Functions and Optimization*, North-Holland, 1991.
11. S. Fujishige and X. Zhang: New algorithms for the intersection problem of submodular systems, *Japan J. Indust. Appl. Math.*, **9** (1992), 369–382.
12. S. Fujishige, H. Röck, and U. Zimmermann: A strongly polynomial algorithm for minimum cost submodular flow problems, *Math. Oper. Res.*, **14** (1989), 60–69.
13. A. V. Goldberg and R. E. Tarjan: A new approach to the maximum flow problem, *J. ACM*, **35** (1988), 921–940.
14. A. V. Goldberg and R. E. Tarjan: Finding minimum-cost circulations by canceling negative cycles, *J. ACM*, **36** (1989), 873–886.
15. M. Grötschel, L. Lovász, and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

16. R. Hassin: Algorithm for the minimum cost circulation problem based on maximizing the mean improvement, *Oper. Res. Lett.*, **12** (1992), 227–233.
17. S. Iwata: A capacity scaling algorithm for convex cost submodular flows, *Math. Programming*, **76** (1997), 299–308.
18. S. Iwata, S. T. McCormick, and M. Shigeno: A faster algorithm for minimum cost submodular flows, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (1998), 167–174.
19. A. V. Karzanov and S. T. McCormick: Polynomial methods for separable convex optimization in unimodular linear spaces with applications, *SIAM J. Comput.*, **26** (1997), 1245–1275.
20. E. L. Lawler and C. U. Martel: Computing maximal polymatroidal network flows, *Math. Oper. Res.*, **7** (1982), 334–347.
21. S. T. McCormick and T. R. Ervolina: Cancelling most helpful total submodular cuts for submodular flow, *Integer Programming and Combinatorial Optimization* (Proceedings of the Third IPCO Conference), G. Rinaldi and L. A. Wolsey eds. (1993), 343–353.
22. S. T. McCormick and T. R. Ervolina: Computing maximum mean cuts, *Discrete Appl. Math.*, **52** (1994), 53–70.
23. S. T. McCormick, T. R. Ervolina and B. Zhou: Mean canceling algorithms for general linear programs and why they (probably) don't work for submodular flow, UBC Faculty of Commerce Working Paper 94-MSC-011 (1994).
24. K. Murota: Discrete convex analysis, *Math. Programming*, **83** (1998), 313–371.
25. K. Murota: Submodular flow problem with a nonseparable cost function, *Combinatorica*, to appear.
26. T. Radzik: Newton's method for fractional combinatorial optimization, *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science* (1992), 659–669; see also: Parametric flows, weighted means of cuts, and fractional combinatorial optimization, *Complexity in Numerical Optimization*, P. Pardalos, ed. (World Scientific, 1993), 351–386.
27. P. Schönsleben: *Ganzzahlige Polymatroid-Intersektions Algorithmen*, Dissertation, Eigenössische Technische Hochschule Zürich, 1980.
28. A. Schrijver: Total dual integrality from directed graphs, crossing families, and sub- and supermodular functions, *Progress in Combinatorial Optimization*, W. R. Pulleyblank, ed. (Academic Press, 1984), 315–361.
29. M. Shigeno, S. Iwata, and S. T. McCormick: Relaxed most negative cycle and most positive cut canceling algorithms for minimum cost flow, *Math. Oper. Res.*, submitted.
30. É. Tardos: A strongly polynomial minimum cost circulation algorithm, *Combinatorica*, **5** (1985), 247–255.
31. C. Wallacher and U. Zimmermann: A polynomial cycle canceling algorithm for submodular flows, *Math. Programming*, to appear.
32. U. Zimmermann: Negative circuits for flows and submodular flows, *Discrete Appl. Math.*, **36** (1992), 179–189.

Edge-Splitting Problems with Demands

Tibor Jordán *

BRICS**, Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
8000 Aarhus C, Denmark
jordan@daimi.au.dk

Abstract. Splitting off two edges su, sv in a graph G means deleting su, sv and adding a new edge uv . Let $G = (V + s, E)$ be k -edge-connected in V ($k \geq 2$) and let $d(s)$ be even. Lovász [8] proved that the edges incident to s can be split off in pairs in a such a way that the resulting graph remains k -edge-connected. In this paper we investigate the existence of such complete splitting sequences when the set of split edges has to satisfy some additional requirements. We prove structural properties of the set of those pairs u, v of neighbours of s for which splitting off su, sv destroys k -edge-connectivity. This leads to a new method for solving problems of this type.

By applying this new approach first we obtain a short proof for a recent result of Nagamochi and Eades [9] on planarity-preserving complete splitting sequences.

Then we apply our structural result to prove the following: let G and H be two graphs on the same set $V + s$ of vertices and suppose that their sets of edges incident to s coincide. Let G (H) be k -edge-connected (l -edge-connected, respectively) in V and let $d(s)$ be even. Then there exists a pair su, sv which is allowed to split off in both graphs simultaneously provided $d(s) \geq 6$. If k and l are both even then such a pair exists for arbitrary even $d(s)$. Using this result and the polymatroid intersection theorem we give a polynomial algorithm for the problem of simultaneously augmenting the edge-connectivity of two graphs by adding a smallest (common) set of new edges.

1 Introduction

Edge-splitting is a well-known and useful method to solve graph problems which involve certain edge-connectivity properties. Splitting off two edges su, sv means deleting su, sv and adding a new edge uv . Such an operation may reduce the local edge-connectivity between some pairs of vertices and may decrease the global

* Part of this work was done while the author visited the Department of Applied Mathematics and Physics, Kyoto University, supported by the Monbusho International Scientific Research Program no. 09044160.

** Basic Research in Computer Science, Centre of the Danish National Research Foundation.

edge-connectivity of the graph. The essence of the edge splitting method is to find pairs of edges which can be split off maintaining certain edge-connectivity properties of the graph. If such a good pair exists then one may reduce the problem to a smaller graph which can lead to inductive proofs. Another typical application is the edge-connectivity augmentation problem where splitting off is an important subroutine in some polynomial algorithms. This connection will be discussed in detail in Section 5.

Let $G = (V + s, E)$ be a graph which is k -edge-connected in V , that is, $d(X) \geq k$ holds for the degree of every $\emptyset \neq X \subset V$. Suppose that $d(s)$ is even and $k \geq 2$. Lovász [8] proved that for every edge su of G there exists an edge sv for which splitting off the pair su, sv maintains k -edge-connectivity in V . We call such a pair *admissible*. By repeated applications of this theorem we can see that all the edges incident to s can be split off in pairs in such a way that the resulting graph (on vertex set V) is k -edge-connected. Such a splitting sequence which isolates s (and preserves k -edge-connectivity in V) is called a *complete (admissible) splitting at s* .

This result gives no information about the structure of the subgraph (V, F) induced by the set F of new edges that we obtain by a complete admissible splitting (except the degree-sequence of its vertices, which is clearly uniquely determined by G and is the same for every complete splitting). Recent problems in edge-connectivity augmentation gave rise to edge-splitting problems where the goal is to find a complete admissible splitting for which the subgraph of the new edges satisfies some additional requirement.

The goal of this paper is to develop a new method for these kind of edge splitting problems, to prove new structural results and to give a polynomial algorithm for a new optimization problem.

The basic idea is to define the *non-admissibility graph* $B(s)$ of $G = (V + s, E)$ on the set of neighbours of s by connecting two vertices x, y if and only if the pair sx, sy is not admissible. We prove that $B(s)$ is 2-edge-connected if and only if $B(s)$ is a cycle, k is odd and G has a special structure which we call round. This structural property turns out to be essential in several edge-splitting problems. Suppose that the additional requirement that the split edges have to meet can be given by defining a *demand graph* $D(s)$ on the neighbours of s at every iteration and demanding that only admissible pairs su, sv ($u \neq v$) with $uv \in E(D(s))$ are allowed to split. Then it is clear that such an allowed pair exists if and only if $D(s)$ is not a subgraph of $B(s)$. Our method is to compare the structure of the graphs $B(s)$ and $D(s)$ that may occur at some iteration and show the existence of a complete admissible splitting satisfying the additional requirement by showing that $D(s)$ can never be the subgraph of the corresponding $B(s)$.

As a first application, we show how this new method leads to simple proofs of previous results due to Nagamochi and Eades [9] on planarity-preserving complete admissible splittings and some results of Bang-Jensen, Gabow, Jordán and Szigeti [1] on partition-constrained complete admissible splittings. Then we use our structural results to prove the following new “intersection theorem” for admissible splittings: let two graphs $G = (V + s, E)$ and $H = (V + s, K)$ be given,

for which the two sets of edges incident to s in G and H coincide. Let G and H be k - and l -edge-connected in V , respectively. Then there exists a pair of edges su, sv which is admissible in G and H (with respect to k and l , respectively) simultaneously, provided $d(s) \geq 6$. This property may not hold if $d(s) = 4$ but assuming k and l are both even we prove that a simultaneously admissible pair always exists, showing that a simultaneously admissible complete splitting exists as well.

Using these splitting results and the g -polymatroid intersection theorem we give a min-max theorem and a polynomial algorithm for the *simultaneous edge-connectivity augmentation problem*. In this problem two graphs $G' = (V, E')$, $H' = (V, K')$ and two integers k and l are given and the goal is to find a smallest set of edges whose addition makes G' (and H') k -edge-connected (l -edge-connected, respectively) simultaneously. Our algorithm finds an optimal solution if k and l are both even. In the remaining cases the size of the solution does not exceed the optimum by more than one.

1.1 Definitions and Notation

Graphs in this paper are undirected and may contain parallel edges. Let $G = (V, E)$ be a graph. A *subpartition* of V is a collection of pairwise disjoint subsets of V . The subgraph of G induced by a subset X of vertices is denoted by $G[X]$. A set consisting of a single vertex v is simply denoted by v . An edge joining vertices x and y is denoted by xy . Sometimes xy will refer to an arbitrary copy of the parallel edges between x and y but this will not cause any confusion. Adding or deleting an edge e from a graph G is often denoted by $G + e$ or $G - e$, respectively.

For $X, Y \subseteq V$, $d(X, Y)$ denotes the number of edges with one endvertex in $X - Y$ and the other in $Y - X$. We define the degree of a subset X as $d(X) = d(X, V - X)$. For example $d(v)$ denotes the degree of vertex v . The degree-function of a graph G' is denoted by d' . The set of *neighbours* of v (or v -neighbours, for short), that is, the set of vertices adjacent to vertex v , is denoted by $N(v)$. A graph $G = (V, E)$ is k -edge-connected if

$$d(X) \geq k \quad \text{for all } \emptyset \neq X \subset V. \quad (1)$$

The operation *splitting off* a pair of edges sv, st at a vertex s means replacing sv, st by a new edge vt . If $v = t$ then the resulting loop is deleted. We use $G_{v,t}$ to denote the graph obtained after splitting off the edges sv, st in G (the vertex s will always be clear from the context). A *complete splitting* at a vertex s (with even degree) is a sequence of $d(s)/2$ splittings of pairs of edges incident to s .

2 Preliminaries

This section contains some basic results. The degree function satisfies the following well-known equalities.

Proposition 1. Let $H = (V, E)$ be a graph. For arbitrary subsets $X, Y \subseteq V$,

$$d(X) + d(Y) = d(X \cap Y) + d(X \cup Y) + 2d(X, Y), \quad (2)$$

$$d(X) + d(Y) = d(X - Y) + d(Y - X) + 2d(X \cap Y, V - (X \cup Y)). \quad (3)$$

In the rest of this section let s be a specified vertex of a graph $G = (V + s, E)$ with degree function d such that $d(s)$ is even and (1) holds. Saying (1) holds in such a graph G means it holds for all $\emptyset \neq X \subset V$. A set $\emptyset \neq X \subset V$ is called *dangerous* if $d(X) \leq k+1$ and $d(s, X) \geq 2$. (Notice that in the standard definition of dangerous sets one does not require property $d(s, X) \geq 2$.) A set $\emptyset \neq X \subset V$ is *critical* if $d(X) = k$. Two sets $X, Y \subseteq V$ are *crossing* if $X - Y, Y - X, X \cap Y$ and $V - (X \cup Y)$ are all nonempty. Again these definitions refer only to subsets of V . Edges sv, st form an *admissible pair* in G if $G_{v,t}$ still satisfies (1). It is easy to see that sv, st is not admissible if and only if some dangerous set contains both t and v .

The statements of the following two lemmas can be proved by standard methods using Proposition 1. Most of them are well-known and appeared explicitly or implicitly in [5] (or later in [1]). The proofs are omitted.

Lemma 1. (a) A maximal dangerous set does not cross any critical set.

(b) If X is dangerous then $d(s, V - X) \geq d(s, X)$.

(c) If k is even then two maximal dangerous sets X, Y which are crossing have $d(s, X \cap Y) = 0$. $\square$

Let t be a neighbour of s . A dangerous set X with $t \in X$ is called a *t-dangerous set*.

Lemma 2. Let v be an s -neighbour. Then exactly one of the following holds:

(o) The pair sv, su is admissible for every edge $su \neq sv$.

(i) There exists a unique maximal v -dangerous set X .

(ii) There exist precisely two maximal v -dangerous sets X, Y . In this case k is odd and we have $d(X) = d(Y) = k + 1$, $d(X - Y) = d(Y - X) = k$, $d(X \cap Y) = k$, $d(X \cup Y) = k + 2$, $d(X \cap Y, V + s - (X \cup Y)) = 1$, $d(s, X - Y) \geq 1$, $d(s, Y - X) \geq 1$, and $d(X \cap Y, X - Y) = d(X \cap Y, Y - X) = (k - 1)/2$. $\square$

An s -neighbour v for which alternative (i) (resp. (ii)) holds is called *normal* (*special*, respectively). Note that if k is even then there exist no special s -neighbours by Lemma 2.

The previous lemmas include all ingredients of Frank's proof [5] for the next splitting off theorem due to Lovász.

Theorem 1. [8] Suppose that (1) holds in $G = (V + s, E)$, $k \geq 2$, $d(s)$ is even and $|N(s)| \geq 2$. Then for every edge st there exists an edge su ($t \neq u$) such that the pair st, su is admissible. $\square$

3 The Structure of Non-admissibility Graphs

In this section let $G = (V + s, E)$ be given which satisfies (1) with respect to some $k \geq 2$. First we introduce the *non-admissibility graph* $B(s) = (N(s), E(B(s)))$ of G (with respect to s) as follows. Let the vertices of $B(s)$ correspond to the neighbours of s and let two vertices u, v , $u \neq v$ of B be adjacent if and only if the pair su, sv is not admissible in G . Notice that while G may contain multiple edges $B(s)$ is always a simple graph. $B(s)$ may consist of one vertex only. By definition, two edges su, sv ($u \neq v$) form an admissible pair in G if and only if $uv \in E(\bar{B}(s))$, that is, uv is an edge of the complement of $B(s)$.

This notion will be useful in problems where we search for admissible pairs sx, sy for which the new edge xy obtained by splitting off sx and sy ($x \neq y$) satisfies some extra property Π . For example, one may ask for an admissible split which does not create parallel edges or whose end vertices are both in a given set W of vertices, etc. Suppose that this extra property can be given by a *demand graph* $D(s) = (N(s), E(D(s)))$ defined on the set of neighbours of s in such a way that xy satisfies Π if and only if $xy \in E(D(s))$. By definition, a demand graph is simple. In the previous examples such a $D(s)$ exists: it is the complement of the subgraph of G induced by the s -neighbours in the first case and the union of a complete graph on W and some isolated vertices in the second case. If Π is given by a demand graph $D(s)$ then a split satisfying Π will be called a $D(s)$ -split, or simply a D -split. It is clear from the definitions that a $D(s)$ -split exists if and only if $D(s)$ and $\bar{B}(s)$ has a common edge. In other words, a $D(s)$ -split does not exist if and only if $D(s)$ is a subgraph of $B(s)$.

Thus by showing that $D(s)$ cannot be the subgraph of $B(s)$ we can guarantee the existence of an admissible split satisfying property Π . In order to use this kind of argument, we prove that $B(s)$ satisfies certain structural properties.

In what follows assume that $d(s)$ is even and $|N(s)| \geq 2$. The definition of $B(s)$ and Theorem 1 imply the following.

Proposition 2. *$B(s)$ has no vertex which is adjacent to all the other vertices of $B(s)$.* $\square$

Now we characterize $B(s)$ in the case when k is even. In this case the structure of $B(s)$ is particularly simple and can be described easily.

Theorem 2. *Suppose that $G = (V + s, E)$ satisfies (1) and k is even. Then $B(s)$ is the disjoint union of (at least two) complete graphs.*

Proof. By Lemma 2 each s -neighbour is either isolated in $B(s)$ or is normal. Hence if su, sv and su, sw ($v \neq w$) are both non-admissible then $v, w \in X_u$ follows, where X_u is the unique maximal u -dangerous set. This implies that the pair sv, sw is also non-admissible. By this transitivity property $B(s)$ is the union of pairwise disjoint complete graphs. $B(s)$ itself cannot be complete by Proposition 2. $\square$

It can be seen that every graph consisting of (at least two) disjoint complete graphs can be obtained as a non-admissibility graph. Theorem 2 implies that if

k is even then $B(s)$ is disconnected. Thus for any connected demand graph $D(s)$ there exists a $D(s)$ -split. This simple observation, which illustrates our proof method, will be used later.

Now let us focus on the case when $k \geq 3$ is odd. In this case we do not give a complete characterization of the non-admissibility graphs but characterize those graphs G for which $B(s)$ is 2-edge-connected. To state our result we need some definitions. In a *cyclic partition* $\mathcal{X} = (X_0, \dots, X_{t-1})$ of V the t partition classes $\{X_0, \dots, X_{t-1}\}$ are cyclically ordered. Thus we use the convention $X_t = X_0$, and so on. In a cyclic partition two classes X_i and X_j are *neighbouring* if $|j-i| = 1$ and *non-neighbouring* otherwise. We say that $G' = (V', E')$ is a C_l^p -graph for some $p \geq 3$ and some even $l \geq 2$ if there exists a cyclic partition $\mathcal{Y} = (Y_0, \dots, Y_{p-1})$ of V' for which $d'(Y_i) = l$ ($0 \leq i \leq p-1$) and $d(Y_i, Y_j) = l/2$ for each pair Y_i, Y_j of neighbouring classes of $\mathcal{Y}$ (which implies $d(Y_{i'}, Y_{j'}) = 0$ for each pair of non-neighbouring classes $Y_{i'}, Y_{j'}$). A cyclic partition of G' with these properties is called *uniform*.

Let $G = (V + s, E)$ satisfy (1) for some odd $k \geq 3$. Such a G is called *round* (from vertex s) if $G - s$ is a $C_{k-1}^{d(s)}$ -graph. Note that by (1) this implies that $d(s, V_i) = 1$ for each class V_i ($0 \leq i \leq d(s) - 1$) of a uniform partition $\mathcal{V}$ of $G - s$.

The following lemma will not be used in the proof of the main result of this section but will be used later in the applications. We omit the proof.

Lemma 3. *Let $G = (V + s, E)$ satisfy (1) for some odd $k \geq 3$. Suppose that G is round from s and let $\mathcal{V} = (V_0, \dots, V_r)$ be a uniform partition of V , where $r = d(s) - 1$. Then*

(a) *$G - s$ is $(k-1)$ -edge-connected and for every $X \subset V$ with $d_{G-s}(X) = k-1$ either $X \subseteq V_i$ or $V - X \subseteq V_i$ holds for some $0 \leq i \leq r$ or $X = \bigcup_{i=0}^{i+j} V_i$ for some $0 \leq i \leq r, 1 \leq j \leq r-1$.*

(b) *For any set I of r edges which induces a spanning tree on $N(s)$ the graph $(G - s) + I$ is k -edge-connected.*

(c) *The uniform partition of $G - s$ is unique.*

(d) *$B(s)$ is a cycle on $d(s)$ vertices (which follows the ordering of $\mathcal{V}$).* □

The main structural result is the following.

Theorem 3. *Suppose that $G = (V + s, E)$ satisfies (1), $d(s)$ is even, $|N(s)| \geq 2$ and $k \geq 3$ is odd. Then $B(s)$ is 2-edge-connected if and only if G is round from s and $d(s) \geq 4$.*

Proof. First suppose that G is round from s and $d(s) \geq 4$. Lemma 3(d) shows $B(s)$ is a cycle and hence $B(s)$ is 2-edge-connected.

In what follows we prove the other direction and assume that $B(s)$ is 2-edge-connected. If there is no special s -neighbour then the proof of Theorem 2 shows that $B(s)$ is disconnected. Thus there exists at least one special s -neighbour.

Lemma 4. *Suppose that t is special and let X and Y be the two maximal t -dangerous sets. Let u be a special s -neighbour in $Y - X$. Let Y and Z denote the two maximal u -dangerous sets in G and let su, sv be a non-admissible pair with $v \in Z - Y$. Then $Z \cap X = \emptyset$, $Y = (X \cap Y) \cup (Z \cap Y)$ and $d(s, Y) = 2$.*

Proof. Notice that for every special vertex $u' \in Y - X$ one of the two maximal u' -dangerous sets must be equal to Y , thus Y is indeed one of the two maximal u -dangerous sets.

First suppose $v \in X - Y$. Since X and Y are the only maximal t -dangerous sets we have $t \notin Z$. This shows that v is also special and the two maximal v -dangerous sets are X and Z . Lemma 2 shows $d(Z) = k + 1$ and $d(X - Y) = d(Y - X) = k$. Hence by (3), applied to Z and $X - Y$ and to Z and $Y - X$, we obtain $X - Y \subset Z$ and $Y - X \subset Z$. By Lemma 2 $d(X \cap Y) = k$. Hence $Z \cap X \cap Y = \emptyset$ by Lemma 1, provided $Z \cup (X \cap Y) \neq V$. Moreover, $Z \cup (X \cap Y) = V$ implies $d(s, Z) \geq d(s) - 1 > d(s, V - Z)$, using $d(s) \geq 4$. Since Z is dangerous, this contradicts Lemma 1(b). Thus we conclude $Z \cap X \cap Y = \emptyset$.

The above verified properties of Z and Lemma 2 imply that $k + 1 = d(Z) \geq d(X \cap Y, Z) + d(s, Z) \geq d(X \cap Y, X - Y) + d(X \cap Y, Y - X) + 2 = k - 1 + 2 = k + 1$. This shows that $d(s, Z) = 2$ and hence $d(s, Z \cup (X \cap Y)) = 3$ holds. We also get $d(Z, V - Z - (X \cap Y)) = 0$ and by Lemma 2 we have $d(X \cap Y, V - Z - (X \cap Y)) = 0$ as well. Therefore $d(Z \cup (X \cap Y), V - Z - (X \cap Y)) = 0$ and hence $d(s, Z \cup (X \cap Y)) = d(Z \cup X \cup Y) = 3 \leq k$. This shows $Z \cup X \cup Y$ is dangerous, contradicting the maximality of X .

Thus we may assume that $v \in V - (X \cup Y)$. By Lemma 2 we have $d(Z) = k + 1$, $d(X - Y) = d(Y - X) = d(X \cap Y) = k$ and there exists an s -neighbour $w \in X - Y$. Clearly, $t \notin Z$ and by the previous argument we may assume $w \notin Z$. We claim that $Z \cap (X - Y) = \emptyset$. Indeed, otherwise Z and $X - Y$ would cross (observe that $t \notin Z \cup (X - Y)$), contradicting Lemma 1(a). We claim that $Z \cap (X \cap Y) = \emptyset$ holds as well. This claim follows similarly: since $w \notin Z \cup (X \cap Y)$, Lemma 1(a) leads to a contradiction.

A third application of Lemma 1(a) shows $Y - X \subset Z$. To see this observe that $t \notin Z \cup (Y - X)$ and hence $(Y - X) - Z \neq \emptyset$ would imply that Z and $Y - X$ cross, a contradiction.

Summarizing the previous observations we obtain $Z \cap X = \emptyset$ and $Y = (X \cap Y) \cup (Z \cap Y)$. By Lemma 2 this implies $d(s, Y) = 2$. This proves the lemma. $\square$

From Lemma 2 we can see that for every normal s -neighbour v the v -neighbours in $B(s)$ induce a complete subgraph of $B(s)$ and for every special s -neighbour t the t -neighbours in $B(s)$ can be divided into two nonempty parts (depending on whether a t -neighbour belongs to $X - Y$ or $Y - X$ for the two maximal t -dangerous sets X and Y), such that each of them induces a complete subgraph of $B(s)$. By Lemma 4 we obtain that there are no edges between these two complete subgraphs of $B(s)$ and hence this bipartition of the t -neighbours in $B(s)$ into complete subgraphs is unique for every special s -neighbour t .

To deduce further properties of $B(s)$ let us fix a special s -neighbour t and take an s -neighbour r ($r \neq t$) which is not adjacent to t in $B(s)$ (that is, for which st, sr is admissible). Such an r exists by Proposition 2. Let the two maximal t -dangerous sets be X and Y . Since $B(s)$ is 2-edge-connected, there exist two edge-disjoint paths P_1, P_2 from t to r in $B(s)$. Without loss of generality we may assume that for the first edge uv of P_1 which leaves $X \cup Y$ (that is, for which $u \in X \cup Y$ and $v \notin X \cup Y$) we have $u \in Y - X$. Clearly, u is a special

s -neighbour (for otherwise t and v , which are both neighbours of u , would be adjacent in $B(s)$) and one of the two maximal u -dangerous sets is Y . Thus by Lemma 4 we get $d(s, Y) = 2$.

Therefore u is the only neighbour of t in $B(s)$ in $Y - X$ and hence P_2 must contain a t -neighbour in $B(s)$ which corresponds to an s -neighbour in $X - Y$. Let w be the last vertex of P_2 in $X - Y$. As above, we obtain that w is special and $d(s, X) = 2$. Hence t has degree two in $B(s)$ and both neighbours of t in $B(s)$ are special. This argument is valid for each special s -neighbour.

Therefore, since $B(s)$ is 2-edge-connected, each s -neighbour must be special and $B(s)$ must be a cycle. Since each s -neighbour is special, there are no parallel edges incident to s . This shows that $B(s)$ has $d(s)$ vertices. Let $v_0, \dots, v_{d(s)-1}$ denote the vertices of $B(s) = C_{d(s)}$, following the cyclic ordering. Let $V_i = X_{v_i}^1 \cap X_{v_i}^2$ ($0 \leq i \leq d(s) - 1$), where $X_{v_i}^1$ and $X_{v_i}^2$ are the two maximal v_i -dangerous sets in G . Now by Lemma 2(ii) and Lemma 4 it is easy to see that $(V_0, \dots, V_{d(s)-1})$ is a uniform partition of G . This implies that G is round from s , as required. $\square$

4 Applications

In this section we apply Theorems 2 and 3 for proving the existence of complete admissible splittings (or properties of maximal admissible splitting sequences) when the set of split edges has to satisfy some additional requirement.

4.1 Edge-Splitting Preserving Planarity

The following theorem is due to Nagamochi and Eades. The new proof we present here is substantially shorter.

Theorem 4. [9] *Let $G = (V + s, E)$ be a planar graph satisfying (1), where k is even or $k = 3$. Then there exists a complete admissible splitting at s for which the resulting graph is planar.*

Proof. First suppose that we are given a fixed planar embedding of G . Notice that this embedding uniquely determines a cyclic ordering $\mathcal{C}$ of (the edges incident to s and hence) the neighbours of s . Clearly, splitting off a pair su, sv for which u and v are consecutive in this cyclic ordering preserves planarity (and a planar embedding of the resulting graph can easily be obtained without reembedding $G - \{su, sv\}$). Thus to see that a complete admissible splitting exists at s which preserves planarity it is enough to prove that (*) there exists an admissible splitting su, sv for which u and v are consecutive in $\mathcal{C}$.

By repeated applications of (*) we obtain a complete admissible splitting which preserves planarity (and the embedding of $G - s$ as well). The existence of a consecutive admissible splitting can be formulated in terms of a demand graph. We may assume $|N(s)| \geq 3$. Let C be a cycle defined on the neighbours of s following the cyclic ordering $\mathcal{C}$. Clearly, a consecutive admissible splitting exists if and only if there exists a $D(s)$ -split by choosing $D(s) := C$. If k is even

then Theorem 2 and the fact that $D(s)$ is connected (while the non-admissibility graph $B(s)$ is disconnected) shows that (*) holds and hence the proof of Theorem 4 is complete in this case. (Note that during the process of iteratively splitting off consecutive admissible pairs the demand cycle C has to be modified whenever s loses some neighbour w by splitting off the last copy of the edges sw .)

Now consider the case $k = 3$. The above argument and Theorem 3 shows (using the fact that $D(s)$ is 2-edge-connected) that by splitting off consecutive admissible pairs as long as possible either we find a complete admissible splitting which preserves planarity or we get stuck in a graph G' which is round from s and for which $B_{G'}(s) = D_{G'}(s)$ holds. In the latter case we need to reembed some parts of G' in order to complete the splitting sequence. We may assume that s is inside one of the bounded faces F of $G' - s$. Let $V_0, \dots, V_{2m-1}$ be the uniform partition of V in $G' - s$ (where $2m := d_{G'}(s)$) and let v_i be the neighbour of s in V_i ($0 \leq i \leq 2m-1$). By Lemma 3 we can easily see that adding the edges v_0v_m and v_iv_{2m-i} ($1 \leq i \leq m-1$) to $G' - s$ results in a 3-edge-connected graph G'' . Observe that G'' is obtained from G' by a complete admissible splitting and all these edges can be added within F in such a way that in the resulting embedding of G'' every edge crossing involves the edge v_0v_m . To avoid these edge crossings in G'' we “flip” V_0 and/or V_m , that is, reembed the subgraphs induced by V_0 and V_m in such a way that after the flippings both v_0 and v_m occur on the boundary of the unbounded face. Since G' is round and $k = 3$ it is easy to see that this can be done. Then we can connect v_0 and v_m within the unbounded face and obtain a planar embedding of the resulting graph. Thus the proof of Theorem 4 is complete. $\square$

The theorem does not hold if $k \geq 5$ is odd, see [9]. Note that the above proof implies that the graphs obtained by a maximal planarity preserving admissible splitting sequence are round for every odd $k \geq 5$. Moreover, it shows that if $k = 3$ then at most two “flippings” are sufficient.

4.2 Edge-Splitting with Partition Constraints

Let $G = (V + s, E)$ be a graph for which (1) holds and $d(s)$ is even and let $\mathcal{P} = \{P_1, P_2, \dots, P_r\}$, $2 \leq r \leq |V|$ be a prescribed partition of V . In order to solve a more general partition-constrained augmentation problem, Bang-Jensen, Gabow, Jordán and Szigeti [1] investigated the existence of complete admissible splittings at s for which each split edge connects two distinct elements of $\mathcal{P}$. They proved that if k is even and an obvious necessary condition holds (namely, $d(s, P_i) \leq d(s)/2$ for every P_i) then such a complete admissible splitting exists and for odd k they characterized those graphs for which such a complete splitting does not exist.

This partition-constrained edge-splitting problem can also be formulated as an edge-splitting problem with demands. Here the demand graph is a complete multipartite graph, which is either 2-edge-connected or is a star. Thus Theorems 2 and 3 can be applied and for several lemmas from [1] somewhat shorter proofs can be obtained. We omit these proofs and note that, as we will point out, the

partition-constrained edge-splitting problem turns out to be a special case of the ‘simultaneous edge-splitting problem’ that we discuss in detail in Section 5.

5 Simultaneous Edge-Splitting and Edge-Connectivity Augmentation

In this section we deal with the following optimization problem: let $G = (V, E)$ and $H = (V, K)$ be two graphs on the same set V of vertices and let k, l be positive integers. Find a smallest set F of new edges for which $\hat{G} = (V, E + F)$ is k -edge-connected and $\hat{H} = (V, K + F)$ is l -edge-connected. Let us call this the *simultaneous edge-connectivity augmentation problem*. We give a polynomial algorithm which finds an optimal solution if both k and l are even and finds a solution whose size is at most one more than the optimum otherwise. One of the two main parts of our solution is a new splitting theorem which we prove using Theorems 2 and 3.

If $G = H$ (and hence wlog $k = l$) then the problem reduces to finding a smallest set F of edges for which $\hat{G} = (V, E + F)$ is k -edge-connected. This is the well-solved *k -edge-connectivity augmentation problem*. For this problem there exist several polynomial algorithms to find an optimal solution. One approach, which is due to Cai and Sun [2] (simplified and extended later by Frank [5]), divides the problem into two parts: first it extends G by adding a new vertex s and a smallest set F' of edges incident to s such that $|F'|$ is even and $G' = (V + s, E + F')$ satisfies (1) with respect to k . Then in the second part, using Theorem 1, it finds a complete admissible splitting from s in G' . The resulting set of split edges will be an optimal solution for the augmentation problem.

Our goal is to extend this approach for the simultaneous augmentation problem. To do this we have to extend both parts: we need an algorithm which finds a smallest F' incident to s for which $G' = (V + s, E + F')$ and $H' = (V + s, K + F')$ *simultaneously* satisfy (1) with respect to k and l , respectively, and then we have to prove that there exists a complete splitting at s which is *simultaneously* admissible in G' and H' .

Both of these extended problems have interesting new properties. While a smallest F' can be found by a greedy deletion procedure in the k -edge-connectivity augmentation problem, this is not the case in the simultaneous augmentation problem. Moreover, a complete splitting at s which is admissible in G' and H' does not always exist. (To see this let $V = \{a, b, c, d\}$, $E = \{ac, bd\}$, $K = \{ab, bc, cd, da\}$, $F' = \{sa, sb, sc, sd\}$ and let $k = 2$, $l = 3$.) However, as we shall see, a smallest F' can be found in polynomial time by solving an appropriate submodular flow problem. Furthermore, if k and l are both even then the required simultaneous complete admissible splitting does exist (and an ‘almost complete’ splitting sequence can be found otherwise).

5.1 Simultaneous Edge-Splitting

Let us start with the solution of the splitting problem. Let $G = (V + s, E + F)$ and $H = (V + s, K + F)$ be given which satisfy (1) with respect to k and l ,

respectively, and let $|F|$ be even, where F denotes the set of edges incident to s . We say that a pair su, sv is *legal* if it is admissible in G as well as in H . A complete splitting sequence at s is *legal* if the resulting graphs (after deleting s) satisfy (1) with respect to k and l , respectively. Let $d(s) := d_G(s) = d_H(s)$.

Theorem 5. *If $d(s) \geq 6$ then there exists a legal pair su, sv . If k and l are both even then there exists a complete legal splitting at s .*

Proof. The property of being legal can be formulated in terms of a demand graph $D(s)$. Namely, a pair su, sv ($u \neq v$) is legal if and only if su, sv is a $D(s)$ -split in G with respect to $D(s) = \bar{B}_H(s)$. Thus the existence of a legal pair follows if we show that $\bar{B}_H(s)$ cannot be a subgraph of $B_G(s)$. Let $D := \bar{B}_H(s)$ and $A := B_G(s)$. We may assume $|N(s)| \geq 4$. (Otherwise, by Proposition 2, $B_H(s)$ has an isolated vertex and hence D has a vertex which is connected to all the other vertices. A has no such vertex.)

First suppose that k and l are both even. In this case Theorem 2 shows that D is connected (since it is the complement of a disconnected graph) and A is disconnected. This implies that a legal pair exists for arbitrary even $d(s)$. Hence a complete legal splitting exists as well. Now suppose that k is odd and l is even. As above, we can see that D , which is a complete multipartite graph by Theorem 2, is either 2-edge-connected (and is not a cycle) or contains a vertex which is connected to all the other vertices or is a four-cycle. In the first two cases Theorem 3 and Proposition 2 show that D cannot be a subgraph of A . In the last case A is 2-edge-connected if no legal split exists. Theorem 3 shows that this may happen only if $A = C_4$ and G is round. In that case there are no parallel edges incident to s and $d(s) = 4$. This proves the theorem when one of k and l is even.

Finally, suppose that k and l are both odd. In this case the proof is more complicated. First we prove the following.

Lemma 5. *Let $H = (V + s, E)$ satisfy (1) with respect to some odd $k \geq 3$. Let $B := B_H(s)$ be the non-admissibility graph of H and let $D := \bar{B}$. Then one of the following holds:*

- (i) D is 2-edge-connected and D is not a cycle,
- (ii) D has a vertex which is adjacent to all the other vertices,
- (iii) $D = C_4$,
- (iv) B arises from a complete graph K_m ($m \geq 2$) by attaching a path of length two to some vertex of K_m ,
- (v) $B = C_4$.

Proof. If B is 2-edge-connected then Theorem 3 gives $B = C_l$ for some even $l \geq 4$. If $l = 4$ then (v) holds, otherwise (i) holds. Hence we may assume that B is not 2-edge-connected. In what follows suppose that neither (i) nor (ii) holds. Let $S := N(s)$ denote the set of vertices of B and D .

Case I: B is disconnected.

Since (ii) does not hold, S has a bipartition $S = X \cup Y$, $X \cap Y = \emptyset$, for which there are no edges from X to Y in B and $|X|, |Y| \geq 2$. Let $p := |X|$ and

$r := |Y|$. Now D contains a spanning complete bipartite graph $K_{p,r}$ and hence D is 2-edge-connected. If $p \geq 3$ or $r \geq 3$ then (i) holds. Thus $D = C_4$ follows, showing (iii) holds.

Case II: B is connected (and has at least one cut-edge).

First consider the subcase when there exists an $X \subset S$ with $|X|, |S - X| \geq 2$ and $d_B(X) = 1$. Let $p := |X|$ and $r := |S - X|$ and let $e = xv$, $v \in X$, be the unique edge leaving X in B . Now D contains a spanning complete bipartite graph minus one edge. Therefore (i) holds if $p, r \geq 3$ and hence we may assume that at least one of p and r is equal to 2. If $p = r = 2$ then $D = P_4$, showing (iv) holds. In the rest of the investigation of this subcase assume $p = 2, r \geq 3$ holds. Let $X = \{v, w\}$. Since B is connected, the edge vw is present in B . Now $D - x$ is 2-edge-connected, thus if $d_D(x) \geq 2$ then (i) holds. Hence $d_D(x) = 1$ which means x is adjacent to every vertex $y \in S - X - \{x\}$ in B . Since v is not adjacent to any vertex in $S - X - \{x\}$ we can see that x is special and $S - X$ induces a complete graph in B by Lemma 2 and Lemma 4. Notice that v is also special since its neighbours x and w are not adjacent. Thus B can be obtained from the complete graph $B[S - X]$ by attaching xv and vw . This gives (iv).

Now consider the subcase when one of the components we obtain by deleting an arbitrary cut-edge from B is a singleton vertex. This means B arises from a 2-edge-connected subgraph M (possibly M is a single vertex) by attaching some leaves, i.e. vertices of degree one. If two leaves aq, bq are adjacent to the same vertex $q \in M$ then q has three pairwise non-adjacent neighbours in B (using $|S| \geq 4$). Thus the set of neighbours of q in B cannot be the union of two complete graphs, contradicting Lemma 2. Thus each vertex $q \in M$ has at most one leaf adjacent to it. If the total number of leaves is at least 3 then this property implies that D contains a spanning complete bipartite graph $K_{p,r}$, ($p, r \geq 3$) minus some independent edges, where one of the color classes, in addition, induces a complete graph. This shows (i) holds. Thus we may assume that the number of leaves is at most two. Let xu be a leaf (and let yv be the other leaf, if exists) with $u, v \in M$. Now $|M| \geq 2$ holds and as we remarked, M is 2-edge-connected. By Lemma 2 u is special. If u is not adjacent to each of the vertices in M then there exist $z, w \in M$ with $uz, zw \in E(B)$ and $uw \notin E(B)$. Now Lemma 4, applied to the two maximal u -dangerous sets and z shows $d_B(u) = 2$ and $d_M(u) = 1$, contradicting the 2-edge-connectivity of M . Thus u is adjacent to each vertex in M . This contradicts Proposition 2 if ux is the unique leaf, hence the other leaf yv exists. Now Lemma 4, applied to the two maximal u -dangerous sets and v shows $M = \{u, v\}$, contradicting the 2-edge-connectivity of M . This contradiction proves the lemma. $\square$

Recall that to prove the theorem we need to show that D is not a subgraph of A provided $d(s) \geq 6$. Thus we are done if (i),(ii) or (v) holds by Theorem 3 and Proposition 2.

Suppose that $D = C_4$ and D is a subgraph of A . Then A is 2-edge-connected and by Theorem 3 we have $A = C_4$ and $d(s) = 4$. This settles case (iii). Now assume (iv) holds. If $m = 2$ then $D = P_4$ (i.e. D is a path on four vertices). In this case if A contains D then by Proposition 2 either $A = C_4$, in which

case $d(s) = 4$ follows, or $A = P_4$. Observe that Lemma 4 implies that the two inner vertices of A are special in G . Now $B = \bar{D}$ is a P_4 as well. Hence the two inner vertices of B (which are disjoint from the inner vertices of A) are special in H . From this it follows that there are no parallel edges incident to s and $d(s) = 4$. Furthermore, $B = P_4$ implies that there exists a dangerous set X in H which contains the two inner vertices x, y of B . It is easy to see that $V - X$ is also dangerous in H and contains the other two neighbours u, v of s , which correspond to the two (non-adjacent) vertices of B with degree one. Thus u and v should also be adjacent in B , a contradiction. This solves case (iv) when $m = 2$.

Finally assume that (iv) holds with $m \geq 3$ and D is a subgraph of A . Using the notation of Lemma 5 we obtain that the edge vw is not present in A by Proposition 2. Hence all the vertices in $S - X - \{x\}$ are special in G by Lemma 2. If w is normal in G then $A[S - X]$ is complete. Thus each vertex u in $S - X - \{x\}$ is adjacent to all the other vertices in A , contradicting Proposition 2. (Now such a u exists since $r \geq 3$.) Thus w is special in G . By Lemma 2 and Lemma 4 $S - X$ can be partitioned into two non-empty complete subgraphs J and L in A according to the two maximal w -dangerous sets J' and L' in G . Without loss of generality assume that $x \in J$. Let $u \in L$. Now Lemma 4, applied to L' , u and v gives $L = \{u\}$. If there exists a vertex $y \in J - x$ then Lemma 4, applied to J' , y and v gives $J = \{y\}$, a contradiction. Thus we conclude that s has four neighbours $\{v, w, u, x\}$ only, contradicting $m \geq 3$. This completes the proof of the theorem. $\square$

From the above proof we can easily deduce the following.

Corollary 1. *Suppose that k or l is odd, $d(s) = 4$ and there exists no legal pair. Then $B_G(s) = C_4$ or $B_H(s) = C_4$ and hence at least one of G and H is round.* $\square$

Note that a complete splitting sequence which is simultaneously admissible in three (or more) graphs does not necessarily exist, even if each of the edge-connectivity values is even. We also remark that the partition-constrained splitting problem can be reduced to a simultaneous edge-splitting problem where at least one of k and l is even. To see this suppose that an instance of the partition-constrained splitting problem is given as in the beginning of Section 4.2. Let $d_m := \max_i \{d_G(s, P_i)\}$ in $G = (V + s, E + F)$ and let $S := N_G(s)$. Build graph $H = (S + x + s, K + F)$ as follows. For each set $S \cap P_i$ in G let the corresponding set in H induce a $(2d_m)$ -edge-connected graph (say, a complete graph with sufficiently many parallel edges or a singleton). The edges incident to s in G and H coincide. Then from vertex x of H add $2d_m - d_G(s, P_i)$ parallel edges to some vertex of $S \cap P_i$ ($1 \leq i \leq r$). Now H satisfies (1) with respect to $l := 2d_m$. It can be seen that a complete admissible splitting satisfying the partition-constraints in G exists if and only if there exists a complete legal splitting in the pair G, H . (Now the sets of vertices of G and H may be different. However, it is easy to see that the assumption $V(G) = V(H)$ is not essential in the simultaneous edge-splitting problem.) This shows that characterizing the pairs G, H for which a

complete legal splitting does not exist (even if one of k and l is even) is at least as difficult as the solution of the partition-constrained problem [1].

5.2 Simultaneous Edge-Connectivity Augmentation

Let $G = (V, E)$ and $H = (V, K)$ be given for which (1) holds with respect to k and l , respectively. First we show how to find a smallest F' for which the extended graphs $G' = (V + s, E + F')$ and $H' = (V + s, K + F')$ simultaneously satisfy (1) with respect to k and l , respectively. As we will see, this problem turns out to be a special submodular flow problem and hence can be solved in polynomial time (see e.g. Cunningham and Frank [3]).

Let V be a finite ground-set and let $p : 2^V \rightarrow Z \cup \{-\infty\}$ be an integer-valued function for which $p(\emptyset) = 0$. We call p *fully supermodular* if $p(X) + p(Y) \leq p(X \cap Y) + p(X \cup Y)$ holds for every $X, Y \subseteq V$. If p is fully supermodular and monotone increasing then $C(p) := \{x \in R^V : x \geq 0, x(A) \geq p(A) \text{ for every } A \subseteq V\}$ is called a *contra-polymatroid*. A function $p : 2^V \rightarrow Z \cup \{-\infty\}$ is *skew supermodular* if for every $X, Y \subseteq V$ either the above submodular inequality holds or $p(X) + p(Y) \leq p(X - Y) + p(Y - X)$. The next result is due to Frank.

Theorem 6. [5] *Let p be a skew supermodular function. Then $C(p)$ is a contra-polymatroid whose unique (monotone, fully supermodular) defining function $\bar{p}$ is given by*

$$\bar{p}(X) := \max \left(\sum_t p(X_i) : \{X_1, \dots, X_t\} \text{ is a subpartition of } X \right). \quad \square$$

Given a graph $G = (V, E)$ and $k \in Z_+$ let $p_G : 2^V \rightarrow Z$ be defined by $p_G(X) := k - d_G(X)$, if $\emptyset \neq X \neq V$ and $p_G(\emptyset) = p_G(V) = 0$. This function p_G is skew-supermodular. Following [5], we say that a vector $z : V \rightarrow Z_+$ is an *augmentation vector* of G if $z(X) \geq p_G(X)$ for every $X \subseteq V$. Observe that $G' = (V + s, E + F)$ satisfies (1) with respect to k if and only if $z(v) := d_F(v)$ ($v \in V$) is an augmentation vector. Hence by Theorem 6 the problem of finding a smallest F can be solved by finding an integer valued element of the contra-polymatroid $C(p_G)$ for which $z(V)$ is minimum. This can be done by a greedy algorithm. Similarly, F' is simultaneously good for G and H if and only if $z(X) \geq \max\{p_G(X), p_H(X)\}$ for every $X \subseteq V$, where $z(v) := d_{F'}(v)$, ($v \in V$). Let us call such a z a *common augmentation vector* of G and H . Observe that finding a common augmentation vector is equivalent to finding an integer valued $z \in C(p_G) \cap C(p_H)$ for which $z(V)$ is minimum. By the *g-polymatroid intersection theorem* (see Frank and Tardos [7] and [4]) the system $\mathcal{S}(p_G, p_H) = \{x \in R^V : x \geq 0, x(A) \geq \max\{p_G(A), p_H(A)\}\}$ is a submodular flow system and hence we can find an integer valued $x \in \mathcal{S}(p_G, p_H)$ minimizing $x(V)$ in polynomial time.

Summarizing the previous results we obtain the following algorithm for the simultaneous edge-connectivity augmentation problem. Let $G = (V, E)$ and $H = (V, K)$ (satisfying (1) with respect to k and l , resp.) be the pair of input graphs.

(Step 1) Find a common augmentation vector for G and H for which $z(V)$ is as small as possible.
 (Step 2) Add a new vertex s to each of G and H and $z(v)$ parallel edges from s to v for every $v \in V$. If $z(V)$ is odd then add one more edge sw for some $w \in V$.
 (Step 3) Find a maximal legal splitting sequence $\mathcal{S}$ at s in the resulting pair of graphs. If $\mathcal{S}$ is complete, let the solution F consist of the set of split edges. Otherwise splitting off $\mathcal{S}$ results in a pair of graphs G', H' for which the degree of s is four. In this case delete s and add a (common) set I of three properly chosen edges to $G' - s$ and $H' - s$. Let the solution F be the union of the split edges and I .

The following theorem shows the correctness of the above algorithm and proves that the solution set F is (almost) optimal. Let us define

$$\Phi_{k,l}(G, H) = \max \left\{ \sum_1^r (k - d_G(X_i)) + \sum_{r+1}^t (l - d_H(X_i)) : \{X_1, \dots, X_t\} \text{ is a subpartition of } V; 0 \leq r \leq t \right\}.$$

The size of a smallest simultaneous augmenting set for G and H (with respect to k and l , resp.) is denoted by $OPT_{k,l}(G, H)$.

Theorem 7. $\lceil \Phi_{k,l}(G, H)/2 \rceil \leq OPT_{k,l}(G, H) \leq \lceil \Phi_{k,l}(G, H)/2 \rceil + 1$. If k and l are both even then $OPT_{k,l}(G, H) = \lceil \Phi_{k,l}(G, H)/2 \rceil$ holds.

Proof. It is easy to see that $\lceil \Phi_{k,l}(G, H)/2 \rceil \leq OPT_{k,l}(G, H)$ holds. We will show that the above algorithm results in a simultaneous augmenting set F with size at most $\lceil \Phi_{k,l}(G, H)/2 \rceil + 1$ (and with size $\lceil \Phi_{k,l}(G, H)/2 \rceil$, if k and l are both even). From the g -polymatroid intersection theorem and our remarks on common augmentation vectors it can be verified that for the vector z that we obtain in Step 1 of the above algorithm we have $z(V) = \Phi_{k,l}(G, H)$ and hence we have $2\lceil \Phi_{k,l}(G, H)/2 \rceil$ edges incident to s at the end of Step 2. By Theorem 5 we can find a maximal sequence of legal splittings in Step 3 which is either complete or results in a pair of graphs G', H' , where the degree of s is four. In the former case the set F of split edges, which is clearly a feasible simultaneous augmenting set, has size $\lceil \Phi_{k,l}(G, H)/2 \rceil$, and hence is optimal. If k and l are both even then such a complete legal splitting always exists, proving $OPT_{k,l}(G, H) = \lceil \Phi_{k,l}(G, H)/2 \rceil$. In the latter case by Corollary 1 one of G' and H' , say G' , is round. Now there exists a complete admissible splitting in H' by Theorem 1. Let $e = uv, f = xy$ be the two edges obtained by such a complete splitting. Let $g = vx$. By Lemma 3(b) adding the edge set $I := \{e, f, g\}$ to G' yields a k -edge-connected graph. Thus the set of edges F which is the union of the edges obtained by the maximal legal splitting sequence and the edge set I is a simultaneous augmenting set. Now $|F| = \lceil \Phi_{k,l}(G, H)/2 \rceil + 1$, as required. $\square$

There are examples showing $OPT = \Phi + 1$ may hold if one of k and l is odd. It is easy to see that the above algorithm can be implemented in polynomial time. (As we pointed out, Step 1 is a submodular flow problem and hence can be solved in polynomial time. One approach to solve Step 3 efficiently is using max-flow computations to check whether a pair of edges is legal or not.) We omit the algorithmic details from this version.

References

1. J. Bang-Jensen, H.N. Gabow, T. Jordán and Z. Szigeti, Edge-connectivity augmentation with partition constraints, Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 1998, pp. 306-315. To appear in SIAM J. Discrete Mathematics.
2. G.R. Cai and Y.G. Sun, The minimum augmentation of any graph to a k -edge-connected graph, Networks 19 (1989), 151–172.
3. W.H. Cunningham, A. Frank, A primal-dual algorithm for submodular flows, Mathematics of Operations Research, 10 (1985) 251-262.
4. J. Edmonds, R. Giles, A min-max relation for submodular functions on graphs, Annals of Discrete Mathematics 1 (1977) 185-204.
5. A. Frank, Augmenting graphs to meet edge-connectivity requirements, SIAM J. Discrete Mathematics, 5 (1992) 22–53.
6. A. Frank, Connectivity augmentation problems in network design, in: Mathematical Programming: State of the Art 1994, (Eds. J.R. Birge and K.G. Murty), The University of Michigan, Ann Arbor, MI, 34-63, 1994.
7. A. Frank and É. Tardos, Generalized polymatroids and submodular flows, Mathematical Programming 42, 1988, pp 489-563.
8. L. Lovász, Combinatorial Problems and Exercises, North-Holland, Amsterdam, 1979.
9. H. Nagamochi and P. Eades, Edge-splitting and edge-connectivity augmentation in planar graphs, R.E. Bixby, E.A. Boyd, and R.Z. Rios-Mercado (Eds.): IPCO VI LNCS 1412, pp. 96-111, 1998., Springer-Verlag Berlin Heidelberg 1998.

Integral Polyhedra Associated with Certain Submodular Functions Defined on 012-Vectors

Kenji Kashiwabara, Masataka Nakamura, and Takashi Takabatake

Graduate Division of International and Interdisciplinary Studies,
University of Tokyo
Komaba, Meguro-ku, Tokyo 153-8902, Japan
{kashiwa,nakamura,takashi}@klee.c.u-tokyo.ac.jp

Abstract. A new class of polyhedra, named greedy-type polyhedra, is introduced. This class contains polyhedra associated with submodular set functions. Greedy-type polyhedra are associated with submodular functions defined on 012-vectors and have 012-vectors as normal vectors of their facets. The face structure of greedy-type polyhedra is described with maximal chains of a certain partial order defined on 012-vectors. Integrality of polyhedra associated with integral greedy-type functions is shown through total dual integrality of the systems of inequalities defining polyhedra. Then a dual algorithm maximizing linear functions over these polyhedra is proposed. It is shown that feasible outputs of certain bipartite networks with gain make greedy-type polyhedra. A separation theorem for greedy-type functions is also proved.

1 Introduction

J. Edmonds and R. Giles introduced the notion of total dual integrality of systems of inequalities [1]. Several combinatorial optimization problems can be formulated as linear programming problems, the sets of whose feasible solutions are described by totally dual integral systems of inequalities [9], [10]. Such systems of linear inequalities have in common a part derived from submodular or supermodular set functions. Linear inequalities defined by set functions determine hyperplanes whose normal vectors are 01-vectors. Thus facets of feasible polyhedra for such problems have only 01-vectors as their normal vectors.

K. Murota, in his theory of “discrete convex analysis” [7], introduced functions on integer lattice points, called M-convex and L-convex functions. An L-convex function is a generalization of a submodular set function and satisfies submodularity on integer lattice points:

$$f(p) + f(q) \geq f(p \vee q) + f(p \wedge q) \text{ for any } p, q \in \mathbb{Z}^S. \quad (1)$$

He proved duality theorems for L- and M-convex functions based on the discrete separation theorem by A. Frank [2] and showed that submodularity may be regarded as a discrete version of convexity.

While L-convex functions satisfy (1), the submodularity of L-convex functions is essentially that of set functions. (It corresponds to the fact that the effective

domain of an M-convex function makes a polyhedron, normal vectors of whose facets are 01-vectors. See [7].) We would like to investigate polyhedra reflecting submodularity on integer lattice points, facets of which polyhedra may have “non 01-vectors” as their normal vectors. And then we would like to discuss total dual integrality and convexity of such polyhedra. In the first step, we study polyhedra described with submodular functions defined on 012-vectors, which make totally dual integral systems of inequalities.

2 Preliminaries

Throughout this paper, S denotes a nonempty finite set. For a set X , the symbol χ^X denotes the characteristic vector of $X \subseteq S$. For an element $e \in S$, the characteristic vector of e is denoted by χ_e . For a vector $p \in \mathbb{R}^S$, the symbol $\text{supp}(p)$ denotes $\{e \in S \mid p(e) \neq 0\}$ and $\text{supp}^k(p)$ denotes $\{e \in S \mid p(e) = k\}$ for any real k . For vectors $p, q \in \mathbb{R}^S$, the vectors s and t defined by $s(e) := \max\{p(e), q(e)\}$ and $t(e) := \min\{p(e), q(e)\}$ for $e \in S$ are denoted by $p \vee q$ and $p \wedge q$, respectively. The 1-norm of $p \in \mathbb{R}^S$ is $\sum_{e \in S} |p(e)|$ and denoted by $\|p\|$.

The symbol $\mathbb{Z}_+$ ($\mathbb{R}_+$) denotes the set of nonnegative integers (reals), respectively.

A *chain* of a partially ordered set $(S, \preceq)$ is a subset of S which is totally ordered with $\preceq$. A chain is *maximal* if it is a proper subset of no chain. We write $(p_1, \dots, p_m)$ for a chain $\{p_1, \dots, p_m\}$ with $p_i \preceq p_{i+1}$ for $i \in 1, \dots, m-1$.

The *support function* of a convex set $C \subseteq \mathbb{R}^n$ is the function $\delta_C^* : \mathbb{R}^n \rightarrow \mathbb{R}$ defined by $\delta_C^*(x) := \sup\{x^T y \mid y \in C\}$. Any support function f is known to be positively homogeneous [8], that is, $f(kp) = kf(p)$ holds for any positive real k . A set $C \subseteq \mathbb{R}^n$ is called *down-monotone* if for each $y \in S$ all vectors $x \in \mathbb{R}^S$ with $x \leq y$ are in C .

A system of inequalities $Ax \leq b$, where A is an $m \times n$ -matrix and b is an n dimensional vector, is *totally dual integral* (TDI) if (i) A and b are rational and (ii) for each integral vector c such that the LP program $\max\{c^T x \mid Ax \leq b\}$ has the finite optimal value, its dual LP problem $\min\{y^T c \mid yA = b, y \geq \mathbf{0}\}$ has an integral solution. The system is *box TDI* if the system

$$Ax \leq b; \quad l \leq x \leq u$$

is TDI for each pair of rational vectors l and u [1]. It is proved in [1] that if a system of inequalities $Ax \leq b$ is TDI with an integral vector b , then $\{x \in \mathbb{R}^S \mid Ax \leq b\}$ is an integral polyhedron.

A set function $f : 2^S \rightarrow \mathbb{R}$ is *submodular* if it satisfies $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ for any $X, Y \subseteq S$. We assume $f(\emptyset) = 0$ in the following.

A polyhedron $P(f)$ is associated with a submodular set function f on 2^S with $f(\emptyset) = 0$ by $P(f) = \{x \in \mathbb{R}^S \mid \sum_{e \in X} x(e) \leq f(X) \ (X \subseteq S)\}$. Such a polyhedron is known to be TDI and box TDI [1].

A submodular set function f on $2^S \rightarrow \mathbb{R}$ is extended to a convex function $\hat{f}$ on $\mathbb{R}_+^S$ by $\hat{f}(p) := \max\{p^T x \mid x \in P(f)\}$ ($p \in \mathbb{R}_+^S$) in [6]. This extended function

$\hat{f}$ is called the *Lovász extension* of f . It is proved in [7] that $g : \mathbb{Z}^S \rightarrow \mathbb{Z}$ is the restriction of the Lovász extension of an integer-valued submodular set function to $\mathbb{Z}^S$ if and only if g satisfies the following properties:

- (L0) g is positively homogeneous;
- (L1) $g(p) + g(q) \geq g(p \vee q) + g(p \wedge q)$ for any $p, q \in \mathbb{Z}^S$;
- (L2) There exists an integer r such that $g(p + \mathbb{1}) = g(p) + r$ for any $p \in \mathbb{Z}^S$.

It is obvious from (L2) the values of the Lovász extension for any nonnegative vectors are nonnegative combinations of values for 01-vectors.

A function $g : \mathbb{Z}^S \rightarrow \mathbb{Z} \cup \{+\infty\}$ is called *L-convex* if g satisfies (L1) and (L2) and if $g(p) \in \mathbb{Z}$ for some $p \in \mathbb{Z}^S$ [7]. The class of L-convex functions thereby contains the restriction of the Lovász extension to $\mathbb{Z}^S$ of any integer-valued submodular set function.

3 Definition of Greedy-Type Polyhedra

In this section we introduce a class of polyhedra, named greedy-type polyhedra, defined with functions on 012-vectors.

Definition 1 (Greedy-type function). *Let S be a nonempty finite set. We call a function $f : \{0, 1, 2\}^S \rightarrow \mathbb{R}$ greedy-type on $\{0, 1, 2\}^S$ if it satisfies the following properties:*

- (G1) $f(2p) = 2f(p)$ for every $p \in \{0, 1\}^S$;
- (G2) $f(p) + f(q) \geq f(p \vee q) + f(p \wedge q)$ for any $p, q \in \{0, 1, 2\}^S$;
- (G3) $f(\chi^X + \chi^Y) + f(\chi^X + \chi^Z) \geq f(\chi^Y + \chi^Z) + f(2\chi^X)$ for any $X, Y, Z \subseteq S$ with $X \subsetneq Y \subsetneq Z$.

We refer simply as (G1), (G2), and (G3) to the respective properties in Definition 1 in what follows.

Remark 2. Property (G1) can be regarded as positive homogeneity and implies $f(\mathbf{0}) = 0$. Property (G2) is submodularity. Property (G3) implies, $f(p) \leq f(\chi^{\text{supp}(p)} + \chi^{\text{supp}^2(p)})$ as its special case. This inequality shows that values of greedy-type functions for non 01-vectors possess non-redundant information, while those of the Lovász extensions of submodular set functions do not.

Remark 3. Restrictions of the Lovász extensions of submodular set functions to $\{0, 1, 2\}^S$ satisfy (G1), (G2), and (G3). They are thereby greedy-type functions. L-convex functions satisfy (G2) but neither (G1) nor (G3), which implies that restrictions of L-convex functions to $\{0, 1, 2\}^S$ are not greedy-type.

On the other hand, an integral greedy-type function may not be the restriction of any L-convex function. An example of such functions, the cardinality of whose support is two, is defined by $f((0, 0)) := 0$, $f((1, 0)) := 10$, $f((0, 1)) := 9$,

$f((1,1)) := 14$, $f((2,1)) := 22$, $f((1,2)) := 22$. But f fails (L2) and is the restriction of no L-convex function. (This function f is not the restriction of any $L^\natural$ -convex function introduced in [4], either.)

Functions on $\{0, \pm 1\}^S$, such as *bisubmodular functions* and functions determining *ternary semimodular polyhedra*, correspond to certain functions on $\{0, 1, 2\}^S$ with a 1 to 1 mapping between $\{0, \pm 1\}$ and $\{0, 1, 2\}$. (See [3] for definitions of bisubmodular functions and ternary semimodular polyhedra.) Functions on $\{0, 1, 2\}^S$ corresponding to bisubmodular functions do not satisfy (G2) with any mapping. On the other hand, there exists a mapping with which functions on $\{0, 1, 2\}^S$ corresponding to those on $\{0, 1\}^S$ determining ternary semimodular polyhedra satisfy (G2). But the greedy-type function f described in the previous paragraph also fails the necessary condition for functions determining ternary semimodular polyhedra.

The focus of this paper is to study polyhedra associated with greedy-type functions in the following way.

Definition 4 (Greedy-type polyhedra). *A polyhedron in $P \subseteq \mathbb{R}^S$ is called a greedy-type polyhedron if there exists a greedy-type function on $\mathbb{R}^S$ such that $P = \{x \in \mathbb{R}^S \mid p^T x \leq f(p) \ (p \in \{0, 1, 2\}^S)\}$. Then the polyhedron P is also denoted by $P(f)$, where f is such a greedy-type function.*

We give examples of such polyhedra and polyhedra associated with submodular set functions in Fig. 1.

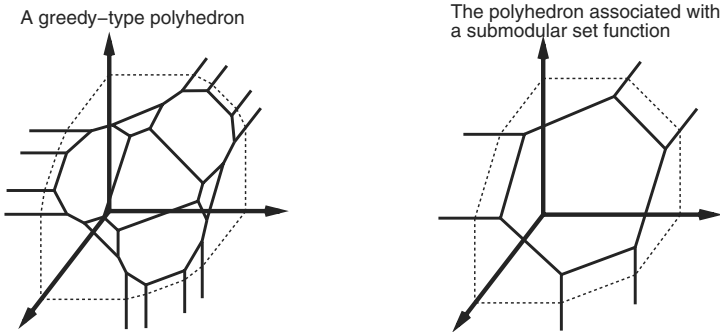


Fig. 1. A greedy-type polyhedron and the polyhedron associated with a submodular set function

4 Face Structure of Greedy-Type Polyhedra

In this section, the face structure of greedy-type polyhedra is described with maximal chains of a poset defined on a subset of 012-vectors. We first make several definitions necessary for the definition of this poset.

We write $\text{Ir}(S)_2$ for $\{0, 1, 2\}^S \setminus \{0, 2\}^S$, which is the set of “irreducible” vectors in $\{0, 1, 2\}^S$.

Now let S consist of n elements $s_1, \dots, s_n$. A permutation σ of $\{1, \dots, n\}$ gives a total order $\leq_\sigma$ on S such that $s_{\sigma(i)} \leq_\sigma s_{\sigma(j)}$ for every i, j with $i \leq j$. We write $\leq_\sigma$ for this total order. For $k_1 \in \{0, 1, \dots, n\}$, we write $\text{vec}_\sigma(k_1)$ for the characteristic vector of the set $\{s_{\sigma(i)} \in S \mid 1 \leq i \leq k_1\}$. The symbol $\text{vec}_\sigma(k_2, k_1)$ denotes $\text{vec}_\sigma(k_2) + \text{vec}_\sigma(k_1)$ in what follows. We sometimes omit σ when there is no danger of confusion.

A vector $p \in \{0, 1, 2\}^S$ is *simply non-increasing under $\leq_\sigma$* if $p(s_{\sigma(i)}) \geq p(s_{\sigma(j)})$ for any $i, j \in \{1, \dots, n\}$ with $i \leq j$.

With these definitions, we introduce a partial order, the set of whose maximal chains is the main issue of this section.

Definition 5 (Partial order $\preceq_2$). We define a partial order $\preceq_2$ on $\text{Ir}(S)_2$ so that $p \preceq_2 q$ if both p and q are simply non-increasing under some total order on S and if

- (a) $p \leq q$ and $\text{supp}(p) = \text{supp}(q)$, or,
- (b) $p \leq q$ and $\text{supp}(p) \subseteq \text{supp}^2(q)$.

The check for well-definedness of this partial order is easy and left for readers. Here we describe maximal chains of the poset $(\text{Ir}(S)_2, \preceq_2)$.

Lemma 6. Let S be a nonempty set which consists of n elements $1, \dots, n$. Then a sequence of vectors $(p_1, \dots, p_k)$ in $\text{Ir}(S)_2$ is a maximal chain of the partial order $\preceq_2$ if and only if

- (I) the number k of vectors equals n , and,
- (II) there exists a total order $\leq_\sigma$ on S such that
 - (i) $p_n = \text{vec}_\sigma(n-1, n)$, and,
 - (ii) p_i is either $\text{vec}_\sigma(i-1) + \chi^{\text{supp}(p_{i+1})}$ or $\text{vec}_\sigma(i-1, i)$ for $i = 1, \dots, n-1$.

Proof. Obviously a sequence of n vectors $(p_1, \dots, p_n)$ satisfying (I) and (II) makes a chain of the partial order. From Definition 5, two vectors in a chain must have different number of “2”’s. Thus this chain is maximal.

We now prove the “only if” part. It is obvious that any chain of $\preceq_2$ consists of at most n vectors. Let q_1, q_2 be arbitrary vectors in $\text{Ir}(S)_2$ with $q_1 \preceq_2 q_2$ and $|\text{supp}^2(q_1)| + 1 < |\text{supp}^2(q_2)|$. Then, for any $e \in \text{supp}^2(q_2) \setminus \text{supp}^2(q_1)$, the vector $q_2 - \chi_e$ is between q_1 and q_2 with respect to $\preceq_2$. Thus any maximal chain consists of exactly n vectors. Inclusion relationship of $\text{supp}^2(p_i)$ ’s determines one unique total order $\leq_\sigma$ on S under which all p_i ’s are simply non-increasing. It is obvious that (i) and (ii) must be satisfied for this total order. $\square$

Corollary 7. Any maximal chain of $(\text{Ir}(S)_2, \preceq_2)$, where S consists of n elements, contain exactly n elements. Let $\leq_\sigma$ be the total order on S under which all vectors of a maximal chain $(p_1, \dots, p_n)$ are simply non-increasing. Then p_k coincides with $\text{vec}_\sigma(k-1, j)$ for some j with $k \leq j \leq n$ and $p_i = \text{vec}_\sigma(i-1, j)$ holds for any i with $k \leq i \leq j$.

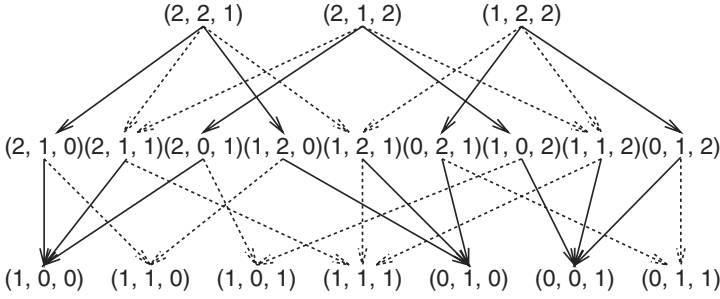


Fig. 2. Maximal chains of $(\text{Ir}(S)_2, \preceq_2)$ for $|S|=3$

Figure 2 shows the set of maximal chains of the partial order $\preceq_2$, where $|S|$ is three.

A maximal chain $(p_1, \dots, p_n)$ of the poset $(\text{Ir}(S)_2, \preceq_2)$ together with a function on $\text{Ir}(S)_2$ determines a point in $\mathbb{R}^S$ as the intersection of hyperplanes $p_i^T x = f(p_i)$ ($i = 1, \dots, n$). We state this fact as the following lemma. We call this point the point corresponding to the maximal chain in what follows.

Lemma 8. *Let $(p_1, \dots, p_n)$ be a maximal chain of a poset $(\text{Ir}(S)_2, \preceq_2)$, where S is a finite set with n elements. Then the set of vectors $\{p_1, \dots, p_n\}$ is linearly independent.*

Proof. We may assume, without loss of generality, that S is composed with $1, \dots, n$ so that $\text{supp}^2(p_i) = \{1, \dots, i-1\}$ for $i \geq 2$. We prove the lemma by showing that the $n \times n$ -matrix $(p_n, \dots, p_1)^T$ is nonsingular.

Let p_1 be $\text{vec}(j)$. By Corollary 7, $p_i = \text{vec}(i-1, j)$ for every $i = 1, \dots, j$. Then $\chi_i = p_{i+1} - p_i$ for $i = 1, \dots, j-1$ and $\chi_j = 2p_1 - p_j$. The proof is completed if $j = n$. Otherwise we can reduce the size of the matrix and repeat the same procedure to prove the lemma. $\square$

We would like to show the fact that any point corresponding to a maximal chain belongs to $P(f)$ if f is a greedy-type function. Thus such points are vertices of $P(f)$. The following two lemmas are used to prove this fact.

Lemma 9. *Let f be a function from a subset of $\mathbb{R}^S$ to $\mathbb{R}$, where S consists n elements. Let $p_1, \dots, p_n$ be linearly independent vectors in $\text{dom}(f)$, that is, $P = (p_1, \dots, p_n)^T$ is a nonsingular $n \times n$ -matrix. Let q be an arbitrary vector in $\text{dom}(f)$. Let $a \in \mathbb{R}^S$ denote $(P^{-1})^T q$. The intersection of hyperplanes $p_i^T x = f(p_i)$ ($i = 1, \dots, n$), which is a point, belongs to the half space $q^T x \leq f(q)$ if and only if $f(q) \geq \sum_{i=1}^n a(i)f(p_i)$.*

Proof. Let b be the n dimensional vector $(f(p_1), \dots, f(p_n))^T$. The intersection of the n hyperplanes is $P^{-1}b \in \mathbb{R}^S$. Our claim follows from $q^T(P^{-1}b) = (q^T P^{-1})b = a^T b = \sum_{i=1}^n a(i)f(p_i)$. $\square$

Lemma 10. *Let f be a greedy-type function on $\{0, 1, 2\}^S$ and let p, q be vectors in $\{0, 1, 2\}^S$ such that either $\text{supp}^1(p) \setminus \text{supp}(q)$ or $\text{supp}^1(q) \setminus \text{supp}(p)$ is not empty. Then $f(p) + f(q) \geq f((p + q) \wedge 2\chi^S) + f(p + q - (p + q) \wedge 2\chi^S)$ holds.*

Proof. If $\text{supp}^1(p) \cap \text{supp}^1(q) = \emptyset$, the inequality of the lemma is nothing but that of (G2). So we may assume that $\text{supp}^1(p) \cap \text{supp}^1(q)$ is not empty. Let r denote $2\chi^{\text{supp}^2(p \vee q)}$. By submodularity, we have

$$\begin{aligned} f(p) + f(q) + f(r) &\geq f(p \vee q) + f(p \wedge q) + f(r) \\ &\geq f(p \vee q) + f((p \wedge q) \vee r) + f((p \wedge q) \wedge r) . \end{aligned} \quad (2)$$

Let X, Y, Z denote $\text{supp}(r), \text{supp}((p \wedge q) \vee r), \text{supp}(p \vee q)$, respectively. Then $(p \wedge q) \vee r = \chi^X + \chi^Y$ and $p \vee q = \chi^X + \chi^Z$. By the assumption that $\text{supp}^1(p) \cap \text{supp}^1(q) \neq \emptyset$, we have $X \subsetneq Y$. Since either $\text{supp}^1(p) \setminus \text{supp}(q)$ or $\text{supp}^1(q) \setminus \text{supp}(p)$ is not empty, we have $Y \subsetneq Z$. By (G3), we have

$$\begin{aligned} f(p \vee q) + f((p \wedge q) \vee r) &\geq f(\chi^X + \chi^Y) + f(\chi^X + \chi^Z) \\ &= f(\chi^Y + \chi^Z) + f(2\chi^X) \\ &= f((p + q) \wedge 2\chi^S) + f(r) . \end{aligned} \quad (3)$$

Since $(p \wedge q) \wedge r = p + q - (p + q) \wedge 2\chi^S$, the inequality of the lemma follows from (2) and (3). $\square$

Now we have reached the point to state the main theorem of this section, which ensures that every maximal chain of the poset $(\text{Ir}(S)_2, \leq_2)$ has its corresponding vertex on $P(f)$ for any greedy-type function f .

Theorem 11. *Let f be a greedy-type function whose support is S . Then its associated polyhedron $P(f)$ contains any point corresponding to a maximal chain of $(\text{Ir}(S)_2, \leq_2)$.*

Proof. Let n be $|S|$. Choose an arbitrary maximal chain $(p_1, \dots, p_n)$. Let $x \in \mathbb{R}^S$ be the vertex corresponding to the chain. Let $\leq_\sigma$ denote the total order on S under which $p_1, \dots, p_n$ are simply non-increasing. We omit the symbol σ for simplicity. Let P denote the matrix $(p_1, \dots, p_n)^T$ and b denote the vector $(f(p_1), \dots, f(p_n))^T$. In this proof, $\text{lst}(p)$ denotes the last coordinate that is not 0, more precisely, $\min\{i \in \{0, 1, \dots, n\} \mid p(j) = 0 \text{ for every } j > i\}$. The symbol $\text{lst}^2(p)$ denotes $\min\{i \in \{0, 1, \dots, n\} \mid p(j) < 2 \text{ for every } j > i\}$.

This theorem is true if x satisfies the linear inequality

$$p^T x \leq f(p) \quad (4)$$

for every $p \in \{0, 1, 2\}^S$. By Lemma 9, (4) is equivalent to

$$f(p) \geq \sum a(i)f(p_i), \quad (5)$$

where a is the only vector in $\mathbb{R}^S$ with $p = \sum_{i=1}^n a(i)p_i$. If $p + p_k = \sum m_i q_i$ holds for (i) some vector p_k in the maximal chain, (ii) some nonnegative integers m_i 's, and (iii) some vectors q_i 's for which (5) is shown, then the following inequality

$$f(p) + f(p_k) \geq \sum m_i f(q_i) \quad (6)$$

implies (5). We thereby prove this theorem by showing (5) or (6) for every $p \in \{0, 1, 2\}^S$.

In this proof we call a vector $p \in \{0, 1, 2\}^S$ hole-less under $\leq_\sigma$ if $\chi^{\text{supp}(p)} = \text{vec}_\sigma(k_1)$ for some $k_1 \in \{0, 1, \dots, n\}$. We say that a vector has a hole if it is not a hole-less vector.

We first show these inequalities for vectors simply non-increasing under $\leq_\sigma$. Then for hole-less vectors in $\{0, 1, 2\}^S$ and lastly for arbitrary vectors in $\{0, 1, 2\}^S$. It should be noted that (5) holds if the vector p belongs to the chain.

We show (5) or (6) for a simply non-increasing vector p , using induction on $\text{lst}(p)$. Inequality (5) is trivial if $\text{lst}(p) = 0$, that is $p = \mathbf{0}$. Suppose that (5) holds for any simply non-increasing vector $p \in \{0, 1, 2\}^S$ with $\text{lst}(p) \leq k-1$. We examine whether (5) or (6) holds for a vector $p \in \{0, 1, 2\}^S$ with $\text{lst}(p) = k$.

We first assume that $p_k = \text{vec}(k-1, k)$. We also assume that neither p nor $\frac{1}{2}p$ belongs to the chain, since (5) is trivial otherwise. By Corollary 7, this assumption implies that $\text{supp}(p_1) \subsetneq \{1, \dots, k\}$. Let p_l be the last vector in the chain with $\text{supp}(p_l) \subsetneq \{1, \dots, k\}$. We see from Corollary 7 that $p_l = \text{vec}(l-1, l)$, $p_{l+1} = \text{vec}(l, k)$, and $\text{supp}^2(p) \subseteq \{1, \dots, l-1\}$ hold. Then $p + p_l = p - \chi^{\{l, \dots, k\}} + p_{l+1}$ holds. By Lemma 10, we have $f(p) + f(p_l) \geq f(p_{l+1}) + f(p - \chi^{\{l, \dots, k\}})$. Since $p - \chi^{\{l, \dots, k\}}$ is a simply non-increasing vector with $\text{lst}(p - \chi^{\{l, \dots, k\}}) < k$, (6) is proved.

Now we assume that $p_k = \text{vec}(k-1, j)$ for some $j > k$. Then $p + p_k = p - \chi_k + p_{k+1}$ holds. By Lemma 10, we have $f(p) + f(p_k) \geq f(p - \chi_k) + f(p_{k+1})$. Since $p - \chi_k$ is a simply non-increasing vector with $\text{lst}(p - \chi_k) < k$, we have shown (6).

We show (6) for a hole-less vector p in $\{0, 1, 2\}^S$, using induction on $\|p\|$. If $\|p\| \leq 2$, then p is a simply non-increasing vector, for which we have already shown (5). Suppose that (5) holds for every hole-less vector $p \in \{0, 1, 2\}^S$ with $\|p\| \leq k-1$. We examine whether (6) holds for a hole-less vector $p \in \{0, 1, 2\}^S$ with $\|p\| = k$. Let l denote $\text{lst}^2(p)$.

We may assume $l \geq 2$ since otherwise p is a simply non-increasing vector. Obviously $p \vee p_l$ is simply non-increasing, and $p \wedge p_l$ is a hole-less vector with $\|p \wedge p_l\| \leq k-1$. Since $f(p) + f(p_l) \geq f(p \vee p_l) + f(p \wedge p_l)$ follows from (G2), we see that (6) holds for such p .

We show (6) for an arbitrary vector p in $\{0, 1, 2\}^S$, using induction on $\|p\|$. If $\|p\| = 0$, then p equals $\mathbf{0}$ and (6) holds for such p . Suppose that (6) holds for every $p \in \{0, 1, 2\}^S$ with $\|p\| \leq k-1$. We examine whether (6) holds for a vector $p \in \{0, 1, 2\}^S$ with $\|p\| = k$.

Let l be “the last coordinate of holes”, more precisely, $\max\{i \in S \setminus \{n\} \mid p(i) = 0 \text{ and } p(i+1) \neq 0\}$. We may assume that p has a hole and then $l \geq 1$ holds. Then $(p + p_l) \wedge 2\chi^S$ is a hole-less vector and $\|p + p_l - (p + p_l) \wedge 2\chi^S\| \leq k-1$. Since $p(l) = 0$

and $p_l(l) = 1$, we have $f(p) + f(p_l) \geq f((p + p_l) \wedge 2\chi^S) + f(p + p_l - (p + p_l) \wedge 2\chi^S)$ by Lemma 10. Thus (6) is shown. $\square$

5 Integrality of Greedy-Type Polyhedra

In this section, we discuss integrality of greedy-type polyhedra. We prove the total dual integrality of the system of linear inequalities:

$$p^T x \leq f(p) \quad (p \in \{0, 1, 2\}^S) \quad (7)$$

for a rational greedy-type function f defined on $\{0, 1, 2\}^S$.

Lemma 12. *Every vector $c \in \mathbb{R}_+^S$ is a nonnegative combination of vectors in a certain maximal chain of the poset $(\text{Ir}(S)_2, \preceq_2)$. Moreover if the vector is integral, the coefficients may be taken from $\mathbb{Z}_+$.*

Proof. Let the set S be $\{1, \dots, n\}$. Considering permutations of S , we only have to show this theorem for simply non-increasing vector c . This theorem is proved as a corollary of the claim stated in the following paragraph.

Let j be some integer in $\{1, \dots, n\}$ and let c be a simply non-increasing vector. Let l denote $\min\{i \in \{j, \dots, n\} \mid c(i') = 0 \text{ } (i' > i)\}$. If both $c(i) = c(j)$ for $i = j + 1, \dots, l$ and $c(i) = 0$ for $i = l + 1, \dots, n$ are satisfied, there exists a maximal chain $(p_1, \dots, p_n)$ of $(\text{Ir}(S)_2, \preceq_2)$ such that (i) c is a nonnegative combination of $p_1, \dots, p_j$ and (ii) $\text{supp}(p_j) = \{1, \dots, l\}$. Moreover if c is integral, the coefficients may be taken from $\mathbb{Z}_+$.

We use induction on j to prove this claim. If $j = 1$, the claim is trivial. Suppose that the claim holds for any integer j with $j \leq k$. We now show the claim for $j = k + 1$.

We consider the following two cases according to the vector c . If $c(k) \leq c(k+1) - c(k)$, let c' denote the vector $c - c(k)\text{vec}(k, l)$. And then c' is nonnegative, non-increasing, and $c'(i) = 0$ for $i = k + 1, \dots, n$. If $c(k) > c(k+1) - c(k)$, let c' denote $c - (c(k+1) - c(k))\text{vec}(k, l)$. And then c' is nonnegative, non-increasing, and $c'(i) = c'(k)$ for $i = k, \dots, l$ and $c'(i) = 0$ for $i = l + 1, \dots, n$.

In either case, by the assumption of induction, c' is represented as a nonnegative combination of vectors $p_1, \dots, p_k$, which make a chain of $(\text{Ir}(S)_2, \preceq_2)$. Thus c can be represented as a nonnegative combination of $p_1, \dots, p_k, \text{vec}(k, l)$. Since either $\text{supp}(p_k) \subseteq \text{supp}^2(\text{vec}(k, l))$ or $\text{supp}(p_k) = \text{supp}(\text{vec}(k, l))$ holds, vectors $p_1, \dots, p_k, \text{vec}(k, l)$ also make a chain.

If c is integral, the coefficient for $\text{vec}(k, l)$ is an integer and the other coefficients are also integral since the vector c' is also integral.

Lemma 12 follows from the claim for $j = n$. $\square$

Corollary 13. *For every vertex of a greedy-type polyhedron in $\mathbb{R}^S$, there exists a maximal chain of $(\text{Ir}(S)_2, \preceq_2)$ corresponding to the vertex.*

Theorem 14. *Let f be a rational greedy-type function defined on $\{0, 1, 2\}^S$, where S is a nonempty finite set. Then the system of linear inequalities*

$$p^T x \leq f(p) \quad (p \in \{0, 1, 2\}^S)$$

is totally dual integral.

Proof. The LP problem $\max\{c^T x \mid p^T x \leq f(p) \ (p \in \{0, 1, 2\}^S)\}$ has a finite optimal value if and only if c is nonnegative. By Lemma 12, a nonnegative vector c_0 is a nonnegative combination of vectors in some maximal chain of the poset $(\text{Ir}(S)_2, \leq_2)$. Thus the coefficients of the combination make an integral feasible solution of the dual problem $\min\{y^T b \mid yA = c_0, y \geq \mathbf{0}\}$, where b is the appropriate vector composed with $f(p)$'s. Let y_0 denote this feasible solution and let x_0 denote the vertex on $P(f)$ corresponding to the chain. It is easy to show $y_0^T b = c_0^T x_0$. By the duality theorem of linear programming, we see that this integral feasible solution of the dual problem is an optimal solution. Thus the system of inequality is TDI. $\square$

Corollary 15. *Let f be an integral greedy-type function defined on $\{0, 1, 2\}^S$, where S is a nonempty finite set. Then its associated polyhedron $P(f)$ is integral.*

Remark 16. Such a system of linear inequalities is not box TDI. Let f be a greedy-type function derived from the polyhedron $\{x \in \mathbb{R}^{\{s_1, s_2, s_3\}} \mid x(s_1) \leq 2, x(s_2) \leq 4, x(s_3) \leq 1, 2x(s_1) + x(s_2) + 2x(s_3) \leq 6\}$. Adding inequalities corresponding to the box $(0, 0, 0)^T \leq x \leq (2, 4, 1)^T$ to the system of linear inequalities $p^T x \leq f(p) \ (p \in \{0, 1, 2\}^{\{s_1, s_2, s_3\}})$ makes a system of inequalities which is not TDI. For example, the objective vector $(3, 2, 1)^T$ is maximized by the vertex $(1, 4, 0)$, while $(3, 2, 1)^T$ is not any nonnegative integral combination of vectors $(2, 1, 2)^T, (0, 0, -1)^T, (0, 1, 0)^T$, which correspond to the vertex $(1, 4, 0)$. (See Fig 3.)

In the end of this section, we propose a dual greedy algorithm which solves the LP problem $\max\{c^T x \mid p^T x \leq f(p) \ (p \in \{0, 1, 2\}^S)\}$ for a greedy-type function $f : \{0, 1, 2\}^S \rightarrow \mathbb{R}$ and a nonnegative objective vector $c \in \mathbb{R}_+^S$. Readers may easily see that Algorithm 17 gives a representation of the objective vector as a nonnegative linear combination of vectors in a maximal chain. Then the validity of Algorithm 17 follows from duality theorem of linear programming and Theorem 11. (It is also proved in [11].)

Algorithm 17 (Dual greedy algorithm for greedy-type functions).

Input A greedy-type function $f : \{0, 1, 2\}^S \rightarrow \mathbb{R}$, where S is a finite set with n elements. A nonnegative objective vector $c \in \mathbb{R}_+^S$.

Output The maximum value of $c^T x$ among $x \in P(f)$. A vertex $\bar{x} \in P(f)$ which gives the maximum value.

Step1 Sort the support $S = \{1, 2, \dots, n\}$ so that $c(i) \geq c(i+1) \ (1 \leq i \leq n-1)$.

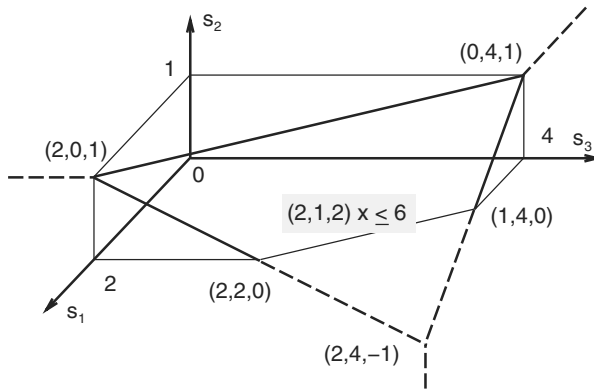


Fig. 3. The polytope corresponding to the system of inequalities in Remark 16

Step2 Let t, y be n dimensional vectors. Initialize $c_{n+1} := c$, $v_{\max} := 0$.

Step3 For $i := n, n-1, \dots, 2, 1$, repeat;

$$\begin{aligned} y(i) &:= \min\{c_{i+1}(i-1) - c_{i+1}(i), c_{i+1}(i)\}, \\ l(i) &:= \begin{cases} i & \text{if } c_{i+1}(i+1) = 0, \\ l(i+1) & \text{otherwise,} \end{cases} \\ p_i &:= \text{vec}(i-1, l(i)), \\ c_i &:= c_{i+1} - y(i)p_i, \\ v_{\max} &:= v_{\max} + y(i)f(p_i), \end{aligned}$$

where $c_{n+1}(n+1)$, $c_2(0)$ is considered to be 0, $+\infty$, respectively.

End Return the value $v_{\max}$ and the vector $P^{-1}b$ as the outputs, where P is the $n \times n$ -matrix with p_i^T as its i th row, and b is the n dimensional vector with $f(p_i)$ as its i th component, for $i = 1, \dots, n$.

6 An Example and Separation Theorem

We mention here some bipartite network flow with gain whose feasible flows make a greedy-type polyhedron. A separation theorem for greedy-type functions is also shown in this section.

Definition 18 (Bipartite network with $\{1, 2\}$ -gain). Let $D = (S^+, S; A)$ be a simple bipartite digraph such that $A = \{(s, t) \mid s \in S^+, t \in S\}$. The members of S^+ are called sources and those of S sinks. Let an upper capacity function $\bar{c} : A \rightarrow \mathbb{R}$ and a supply function $b : S^+ \rightarrow \mathbb{R}$ be given, and a gain function $\alpha : A \rightarrow \{1, 2\}$. We call such a network a bipartite network with $\{1, 2\}$ -gain and write $\mathcal{N} = ((S^+, S; A), \bar{c}, b, \alpha)$ for this network.

Given a bipartite network with $\{1, 2\}$ -gain $\mathcal{N} = ((S^+, S; A), \bar{c}, b, \alpha)$, we define feasible flows of this network as follows. A bipartite flow with $\{1, 2\}$ -gain (or

simply *flow*) of $\mathcal{N}$ is a function $\varphi : A \rightarrow \mathbb{R}$. We define the *generalized boundary* (or simply *boundary*) of a flow φ to be a function $\partial\varphi : S^+ \cup S \rightarrow \mathbb{R}$ defined by $\partial\varphi(v) := \sum_{a \in \delta v^+} \varphi(a) - \sum_{a \in \delta v^-} \alpha(a)\varphi(a)$ for every $v \in S^+ \cup S$. A flow is *feasible* if $\varphi(a) \leq \bar{c}(a)$ for every $a \in A$ and if $\partial\varphi(v) \leq b(v)$ for every $v \in S^+$. We call $-\partial\varphi|_S \in \mathbb{R}^S$ an *output* of $\mathcal{N}$ if φ is a feasible flow of $\mathcal{N}$.

We call the set of all outputs in $\mathbb{R}^S$ of a bipartite network with $\{1, 2\}$ -gain *the output polyhedron* of the network. We call restrictions of the support functions of polyhedra to $\{0, 1, 2\}^S$ *the $\{0, 1, 2\}$ -support functions* of the polyhedra.

From now on, we would like to show that the set of feasible flows makes a greedy-type polyhedron. It is not difficult to prove this statement for bipartite networks with $\{1, 2\}$ -gain whose source sets consist of one element.

Lemma 19. *Let P be the output polyhedron of a bipartite network with $\{1, 2\}$ -gain $\mathcal{N} = ((\{s\}, S; A), \bar{c}, b, \alpha)$, whose source set consists of the only element $\{s\}$. Let f be the $\{0, 1, 2\}$ -support function of P . Then f is a greedy-type function and $P(f) = P$ holds.*

Proof. Let S be $\{1, \dots, n\}$. Let γ be the n dimensional vector defined by $\gamma(i) = 2/\alpha(i)$ for $i \in S$. Obviously $P = \{x \in \mathbb{R}^S \mid \gamma^T x \leq 2b, x(i) \leq \alpha(i)\bar{c}(i) \text{ for any } i \in S\}$ holds.

We first show that f is greedy-type, that is, f satisfies (G1), (G2), and (G3). Since f is a restriction of the support function, it satisfies (G1).

If $b \geq \sum \bar{c}(i)$, then $(\alpha(1)\bar{c}(1), \dots, \alpha(n)\bar{c}(n))$ is the only vertex of P . Since any value of the $\{0, 1, 2\}$ -support function is attained by this vertex, the inequalities of (G2) and (G3) hold with equalities. We now assume $b < \sum \bar{c}(i)$. In this case, P has exactly n vertices, each of which corresponds to the vector set

$$\{\gamma\} \cup \{\chi_j \in \{0, 1\}^S \mid j \in S \setminus \{i\}\}$$

for $i = 1, \dots, n$.

For an arbitrary vector p in $\{0, 1, 2\}^S$, let $\text{argmin}(p, \gamma)$ denote an element l of S that satisfies $p(l)/\gamma(l) \leq p(i)/\gamma(i)$ for any $i \in S$. (If there exists more than one such elements, any choice may be possible for the following argument.) We see that $f(p)$ is attained by the vertex corresponding to vectors in

$$\{\gamma\} \cup \{\chi_j \in \{0, 1\}^S \mid j \in S \setminus \{\text{argmin}(p, \gamma)\}\},$$

and we see that $f(p)$ is equal to

$$\frac{p(\text{argmin}(p, \gamma))}{\gamma(\text{argmin}(p, \gamma))} f(\gamma) + \sum_{j \in S \setminus \{\text{argmin}(p, \gamma)\}} [p(j) - \frac{p(\text{argmin}(p, \gamma))}{\gamma(\text{argmin}(p, \gamma))} \gamma(j)] f(\chi_j) .$$

We write $\text{cntrb}(p)$ for $\frac{p(\text{argmin}(p, \gamma))}{\gamma(\text{argmin}(p, \gamma))}$ in this proof.

Then we have

$$\begin{aligned} f(p) + f(q) - f(p \vee q) - f(p \wedge q) = \\ (\text{cntrb}(p) + \text{cntrb}(q) - \text{cntrb}(p \vee q) - \text{cntrb}(p \wedge q)) \left[\sum_{j \in S} \gamma(j) f(\chi_j) - f(\gamma) \right] . \end{aligned}$$

It is clear that $\sum_{j \in S} \gamma(j)f(\chi_j) - f(\gamma) \geq 0$ and $\text{cntrb}(p) + \text{cntrb}(q) \geq \text{cntrb}(p \vee q) + \text{cntrb}(p \wedge q)$ hold. Thus (G2) is proved and (G3) follows from a similar argument.

Since the normal vectors of P 's facets are 012-vectors, we see that $P = P(f)$ holds. $\square$

Now all we have to prove is that the sum of greedy-type polyhedra is the polyhedron associated with the sum of those functions, which is also a greedy-type function.

Lemma 20. *Let S be a nonempty finite set. Let f and g be greedy-type functions from $\{0, 1, 2\}^S$ to $\mathbb{R}$. Then the vector sum of $P(f)$ and $P(g)$ coincides with $P(f + g)$.*

Proof. Let S contain n elements. Since it is obvious that $P(f + g) \supseteq P(f) + P(g)$, all we have to show is that $P(f + g) \subseteq P(f) + P(g)$.

To prove our claim, we show that $\delta_{P(f+g)}^*(p) \geq \delta_{P(f)+P(g)}^*(p)$ holds for any vector $p \in \mathbb{R}^S$. This inequality is trivial if $p \notin \mathbb{R}_+^S$. (Then both sides are equal to $+\infty$.) Let q be an arbitrary vector in $\mathbb{R}_+^S$. By Lemma 12, q is a nonnegative combination of vectors $p_1, \dots, p_n$ that compose a maximal chain of $(\text{Ir}(S)_2, \preceq_2)$. Let P denote the $n \times n$ -matrix that has p_i as its i th row vector for $i = 1, \dots, n$. Then $q^T = y^T P$ for some nonnegative vector $y \in \mathbb{R}_+^n$. Let b_1 (b_2) be the n dimensional vector that has $f(p_i)$ ($g(p_i)$) as its i th component, respectively. Since the point corresponding to the chain belongs to $P(f + g)$, we have $\delta_{P(f+g)}^*(q) = y^T(b_1 + b_2)$ by the duality theorem of linear programming. Since f and g are greedy-type, $P(f)$ contains $P^{-1}b_1$ and $P(g)$ contains $P^{-1}b_2$. The polyhedron $P(f) + P(g)$ thereby contains $P^{-1}(b_1 + b_2)$. Then $\delta_{P(f)+P(g)}^*(q) \geq q^T P^{-1}(b_1 + b_2) = y^T(b_1 + b_2) = \delta_{P(f+g)}^*(q)$, which we have tried to show, holds. $\square$

As a corollary of Lemma 19 and Lemma 20, we have the following theorem.

Theorem 21. *Let P be the output polyhedron of a bipartite network with $\{1, 2\}$ -gain and let f be the $\{0, 1, 2\}$ -support function of P . Then f is a greedy-type function and $P(f) = P$ holds.*

From Lemma 20, we have Theorem 22, which is a kind of separation theorem.

Theorem 22. *Let S be a nonempty finite set. Let f and g be functions from $\{0, 1, 2\}^S$ to $\mathbb{R}$ such that f and $-g$ are greedy-type. If $f(p) \geq g(p)$ holds for every $p \in \{0, 1, 2\}^S$, there exists a vector $h \in \mathbb{R}^S$ which satisfies $f(p) \geq h^T p \geq g(p)$ for every $p \in \{0, 1, 2\}^S$.*

Proof. We have $P(f - g) = P(f) + P(-g)$ by Lemma 20. Since $f \geq g$, $P(f - g)$ contains the zero vector and so does $P(f) + P(-g)$. It means that there exists a vector $x \in \mathbb{R}^S$ such that $x \in P(f)$ and $-x \in P(-g)$. Then $f(p) \geq x^T p \geq g(p)$ holds for every $p \in \{0, 1, 2\}^S$. $\square$

Remark 23. Property (G3) is essential for Theorem 22. We give two functions defined on $\{0, 1, 2\}^S$ that satisfy (G1) and (G2) and that are not separated by any linear functions.

Let f be the 012-support of the polytope

$$\{x \in \mathbb{R}^{\{s_1, s_2, s_3\}} \mid x(s_1) \leq 2, x(s_2) \leq 4, x(s_3) \leq 1, (2, 1, 2)x \leq 6, (2, 1, 0)x \leq 6\}$$

and let f' be that of

$$\{x \in \mathbb{R}^{\{s_1, s_2, s_3\}} \mid x(s_1) \leq 2, x(s_2) \leq 1, x(s_3) \leq 4, (2, 2, 1)x \leq 6, (2, 0, 1)x \leq 6\} \ .$$

($P(f)$ is the minimum down-monotone polyhedron that includes the nonnegative part of the polytope in Fig. 3. $P(f')$ is obtained from $P(f)$ by exchanging the s_2 -axis and the s_3 -axis.)

Let g be the function on $\{0, 1, 2\}^{\{s_1, s_2, s_3\}}$ defined by $g(p) = (3.5, 1, 4)p - f'(p)$ for $p \in \{0, 1, 2\}^{\{s_1, s_2, s_3\}}$. Then $f \geq -g$ holds. Both f and $-g$ satisfy (G1) and (G2) but not (G3). For example, $f((2, 1, 1)) + f((2, 1, 0))$ is greater than $f((2, 2, 1)) + f((2, 0, 0))$. (See Table 1 for values of f, f', g .)

Since f and g give the same value for three vectors $(2, 0, 1)^T$, $(2, 1, 1)^T$, and $(2, 1, 2)^T$, the only possible vector for h is $(2.5, 1, 0)^T$. But then $f((1, 0, 0))$ is less than $h^T(1, 0, 0)^T$.

Property (G3) is also essential for Lemma 20. The vertex $(3.5, 1, 4)$ of $P(f + f')$ is not contained $P(f) + P(f')$. (The value of $P(f)$'s support function for $(4, 1, 2)^T$ is 10 and that of $P(f')$'s support function is 12, while $(3.5, 1, 4)(4, 1, 2)^T$ is 23.)

The reason why f and g cannot be separated is that the face structures of the polyhedra for f and g' have no common refinement such that normal vectors of its facets are 012-vectors. Property (G3) is necessary for the face structure of greedy-type polyhedra to have such a common refinement. See [5] for relationship between face structures and separation theorem.

Table 1. Values of functions in Remark 23

p	$f(p)$	$f'(p)$	$g(p)$	p	$f(p)$	$f'(p)$	$g(p)$	p	$f(p)$	$f'(p)$	$g(p)$
(0,0,0)	0	0	0	(1,1,1)	5	5	3.5	(2,1,1)	6	6	6
(1,0,0)	2	2	1.5	(2,1,0)	6	5	3	(1,2,1)	9	6	3.5
(0,1,0)	4	1	0	(1,2,0)	9	4	1.5	(1,1,2)	6	9	3.5
(0,0,1)	1	4	0	(0,2,1)	9	6	0	(2,2,1)	10	6	7
(1,1,0)	5	3	1.5	(0,1,2)	6	9	0	(1,2,2)	10	10	3.5
(0,1,1)	5	5	0	(1,0,2)	4	9	2.5	(2,1,2)	6	10	6
(1,0,1)	3	5	2.5	(2,0,1)	5	6	5				

References

[1] Edmonds, J., Giles, R.: A min-max relation for submodular functions on graphs. *Annals of Discrete Mathematics* **1** (1977) 185–204

- [2] Frank, A.: An algorithm for submodular functions on graphs. *Annals of Discrete Mathematics* **16** (1982) 97–120
- [3] Fujishige, S.: Submodular Functions and Optimization. *Annal of Discrete Mathematics* **47** (1991)
- [4] Fujishige, S., Murota, K.: On the relationship between L-convex functions and submodular integrally convex functions. RIMS Preprint 1152, Research Institute for Mathematical Sciences, Kyoto University (1997)
- [5] Kashiwabara, K.: Set Functions and Polyhedra. PhD thesis, Tokyo Institute of Technology (1998)
- [6] Lovász, L.: Submodular functions and convexity. in Grötschel M., Bachem, A., Korte B. (eds.), *Mathematical Programming — The State of the Art*. Springer-Verlag, Berlin (1983) 235–257
- [7] Murota, K.: Discrete convex analysis. *Mathematical Programming.* **83** (1998) 313–371
- [8] Rockafellar, R.T.: *Convex Analysis*. Princeton University Press (1970)
- [9] Schrijver, A.: Total dual integrality from directed graphs, crossing families, and sub- and supermodular functions. in Pulleyblank, W.R. (ed.), *Progress in Combinatorial Optimization*. Academic Press (1984) 315–361
- [10] Schrijver, A.: *Theory of Linear and Integer Programming*. Wiley (1986)
- [11] Takabatake, T.: Generalizations of Submodular Functions and Delta-Matroids. PhD thesis, Universtiy of Tokyo (1998)

Optimal Compaction of Orthogonal Grid Drawings^{*}

(Extended Abstract)

Gunnar W. Klau and Petra Mutzel

Max-Planck-Institut für Informatik
Im Stadtwald, D-66123 Saarbrücken, Germany
{guwek, mutzel}@mpi-sb.mpg.de

Abstract. We consider the two-dimensional compaction problem for orthogonal grid drawings in which the task is to alter the coordinates of the vertices and edge segments while preserving the shape of the drawing so that the total edge length is minimized. The problem is closely related to two-dimensional compaction in VLSI-design and has been shown to be *NP*-hard.

We characterize the set of feasible solutions for the two-dimensional compaction problem in terms of paths in the so-called constraint graphs in x - and y -direction. Similar graphs (known as *layout graphs*) have already been used for one-dimensional compaction in VLSI-design, but this is the first time that a direct connection between these graphs is established. Given the pair of constraint graphs, the two-dimensional compaction task can be viewed as extending these graphs by new arcs so that certain conditions are satisfied and the total edge length is minimized. We can recognize those instances having only one such extension; for these cases we solve the compaction problem in polynomial time.

We transform the geometrical problem into a graph-theoretical one and formulate it as an integer linear program. Our computational experiments show that the new approach works well in practice.

1 Introduction

The compaction problem has been one of the challenging tasks in VLSI-design. The goal is to minimize the area or total edge length of the circuit layout while mainly preserving its shape. In graph drawing the compaction problem also plays an important role. Orthogonal grid drawings, in particular the ones produced by the algorithms based on bend minimization (e.g., [16, 6, 10]), suffer from missing compaction algorithms. In orthogonal grid drawings every edge is represented as a chain of horizontal and vertical segments; moreover, the vertices and bends are placed on grid points. Two orthogonal drawings have the same shape if one can be obtained from the other by modifying the lengths of the horizontal and

^{*} This work is partially supported by the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (No. 03-MU7MP1-4).

vertical segments without changing the angles formed by them. The orthogonal drawing standard is in particular suitable for Entity–Relationship diagrams, state charts, and PERT–diagrams with applications in data bases, CASE–tools, algorithm engineering, work–flow management and many more.

So far, in graph drawing only heuristics have been used for compacting orthogonal grid drawings. Tamassia [16] suggested “refining” the shape of the drawing into one with rectangular faces by introducing artificial edges. If all the faces are rectangles, the compaction problem can be solved in polynomial time using minimum–cost network flow algorithms. In general, however, the solution is far from the optimal solution for the original graph without the artificial edges (see also Sect. 4). Other heuristics are based on the idea of iteratively fixing the x – and y –coordinates followed by a one–dimensional compaction step. In one–dimensional compaction, the goal is to minimize the width or height of the layout while preserving the coordinates of the fixed dimension. It can be solved in polynomial time using the so–called *layout graphs*. The differences of the methods are illustrated in Fig. 1. It shows the output of four different compaction methods for a given graph with fixed shape. The big vertices are internally represented as a face that must have rectangular shape. Figure 1(a) illustrates the method proposed in [16]. Figures 1(b) and 1(c) show the output of two one–dimensional compaction strategies applied to the drawing in Fig. 1(a). Both methods use the underlying layout graphs, the first algorithm is based on longest paths computations, the second on computing maximum flows. Figure 1(d) shows an optimally compacted drawing computed by our algorithm.

The compaction problems in VLSI–design and in graph drawing are closely related but are not the same. Most versions of the compaction problem in VLSI–design and the compaction problem in graph drawing are proven to be NP –hard [11, 14]. Research in VLSI–design has concentrated on one–dimensional methods; only two papers have suggested optimal algorithms for two–dimensional compaction (see [15, 8]). The idea is based on a $(0, 1)$ –non–linear integer programming formulation of the problem and solving it via branch–and–bound. Unfortunately, the methods have not been efficient enough for practical use.

Lengauer states the following: “*The difficulty of two–dimensional compaction lies in determining how the two dimensions of the layout must interact to minimize the area*” [11, p. 581]. The two–dimensional compaction method presented in this paper exactly attacks this point. We provide a necessary and sufficient condition for all feasible solutions of a given instance of the compaction problem. This condition is based on existing paths in the so–called *constraint graphs* in x – and y –direction. These graphs are similar to the *layout graphs* known from one–dimensional compaction methods. The layout graphs, however, are based on visibility properties whereas the constraint graphs arise from the shape of the drawing.

Let us describe our idea more precisely: The two constraint graphs D_h and D_v specify the shape of the given orthogonal drawing. We characterize exactly those extensions of the constraint graphs which belong to feasible orthogonal grid drawings. The task is to extend the given constraint graphs to a complete

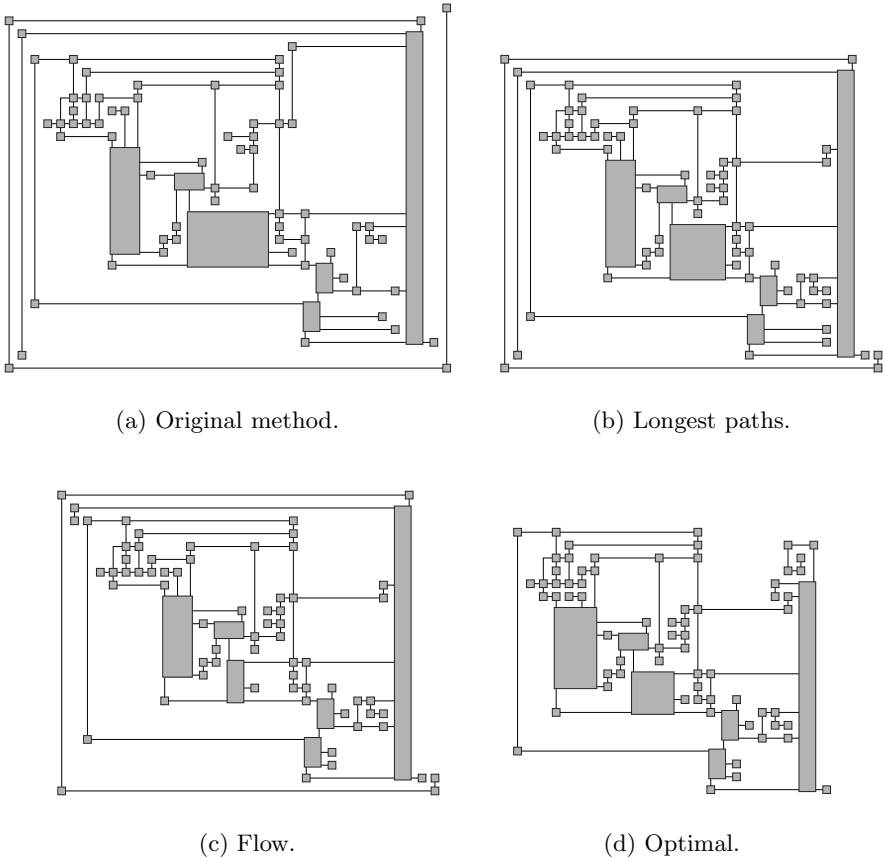


Fig. 1. The result of four different compaction algorithms.

pair of constraint graphs by adding new arcs for which the necessary and sufficient condition is satisfied and the total edge-length of the layout is minimized. Hence, we have transformed the geometrical problem into a graph-theoretical one. Furthermore, we can detect constructively those instances having only one complete extension. For these cases, we can solve the compaction problem in polynomial time.

We formulate the resulting problem as an integer linear program that can be solved via branch-and-cut or branch-and-bound methods. Our computational results on a benchmark set of 11,582 graphs [4] have shown that we are able to solve the two-dimensional compaction problem for all the instances in short computation time. Furthermore, they have shown that often it is worthwhile looking for the optimally compacted drawing. The total edge lengths have been improved up to 37,0% and 65,4% as compared to iterated one-dimensional compaction and the method proposed in [16].

In Sect. 2 we provide the characterization of the set of feasible solutions. The formulation of the compaction problem in the form of an integer linear program is given in Sect. 3. We describe a realization of our approach and corresponding computational experiments on a set of benchmark graphs in Sect. 4. Finally, Sect. 5 contains the conclusions and our future plans. In this version of the paper we have omitted some proofs due to space limitations. They can be found in [9].

2 A Characterization of Feasible Solutions

In this section we describe the transformation of the compaction problem into a graph-theoretical problem. After giving some basic definitions we introduce the notion of shape descriptions and present some of their properties. As a main result of this section, we establish a one-to-one correspondence between shape descriptions satisfying a certain property and feasible orthogonal grid drawings.

2.1 Definitions and Notations

In an *orthogonal (grid) drawing* Γ of a graph G the vertices are placed on mutually distinct integer grid points and the edges on mutually line-distinct paths in the grid connecting their endpoints. We call Γ *simple* if the number of bends and crossings in Γ is zero. Each orthogonal grid drawing can be transformed into a simple orthogonal grid drawing. The straightforward transformation consists of replacing each crossing and bend by a virtual vertex. Figure 2 gives an example.

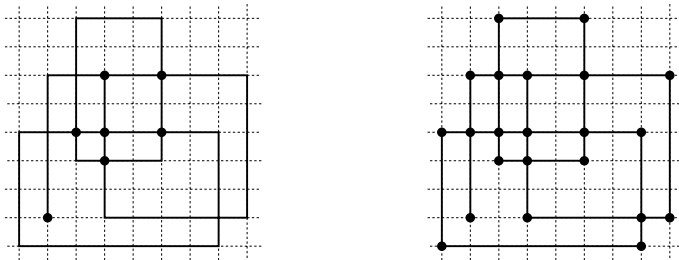


Fig. 2. An orthogonal grid drawing and its simple counterpart.

The shape of a simple drawing is given by the angles inside the faces, i.e., the angles occurring at consecutive edges of a face cycle. Note that the notion of shape induces a partitioning of drawings in equivalence classes. Often, the shape of an orthogonal drawing is given by a so-called *orthogonal representation* H . Formally, for a simple orthogonal drawing, H is a function from the set of faces F to clockwise ordered lists of tuples (e_r, a_r) where e_r is an edge, and a_r is

the angle formed with the following edge inside the appropriate face. Using the definitions, we can specify the compaction problem:

Definition 1. *The compaction problem for orthogonal drawings (COD) is stated by the following: Given a simple orthogonal grid drawing Γ with orthogonal representation H , find a drawing Γ' with orthogonal representation H of minimum total edge length.*

The compaction problem (COD) has been shown to be *NP*-hard [14]. So far, in practice, one-dimensional graph-based compaction methods have been used. But the results are often not satisfying. Figure 3(a) shows an orthogonal drawing with total edge length $2k+5$ which cannot be improved by one-dimensional compaction methods. The reason for this lies in the fact that one-dimensional compaction methods are based on visibility properties. An exact two-dimensional compaction method computes Fig. 3(b) in which the total edge length is only $k+6$.

Let Γ be a simple orthogonal grid drawing of a graph $G = (V, E)$. It induces a partition of the set of edges E into the horizontal set E_h and the vertical set E_v . A horizontal (resp. vertical) *subsegment* in Γ is a connected component in (V, E_h) (resp. (V, E_v)). If the component is maximally connected it is also referred to as a *segment*. We denote the set of horizontal and vertical subsegments by $\mathcal{S}_h$, and $\mathcal{S}_v$ resp., and the sets of segments by $S_h \subseteq \mathcal{S}_h$, $S_v \subseteq \mathcal{S}_v$. We further define $\mathcal{S} = \mathcal{S}_h \cup \mathcal{S}_v$ and $S = S_h \cup S_v$. The following observations are of interest (see also Fig. 4).

1. Each edge is a subsegment, i.e., $E_h \subseteq \mathcal{S}_h$, $E_v \subseteq \mathcal{S}_v$.
2. Each vertex v belongs to one horizontal and one vertical segment, denoted by $\text{hor}(v)$ and $\text{vert}(v)$.
3. Each subsegment s is contained in exactly one segment, referred to as $\text{seg}(s)$.
4. The *limits* of a subsegment s are given as follows: Let v_l, v_r, v_b , and v_t be the leftmost, rightmost, bottommost, and topmost vertices on s . Then $l(s) = \text{vert}(v_l)$, $r(s) = \text{vert}(v_r)$, $b(s) = \text{hor}(v_b)$, and $t(s) = \text{hor}(v_t)$.

The following lemma implies that the total number of segments is $2|V| - |E|$.

Lemma 1. *(proof omitted) Let Γ be a simple orthogonal grid drawing of a graph $G = (V, E_h \cup E_v)$. Then $|S_h| = |V| - |E_h|$ and $|S_v| = |V| - |E_v|$.*

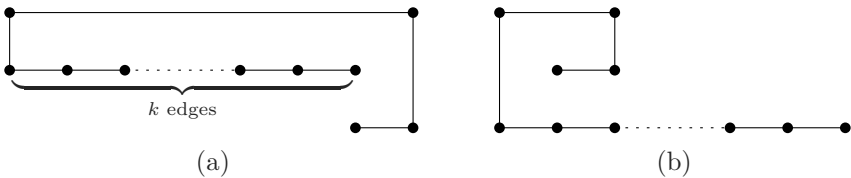


Fig. 3. A drawing generated with (a) an iterative one-dimensional and (b) an optimal compaction method.

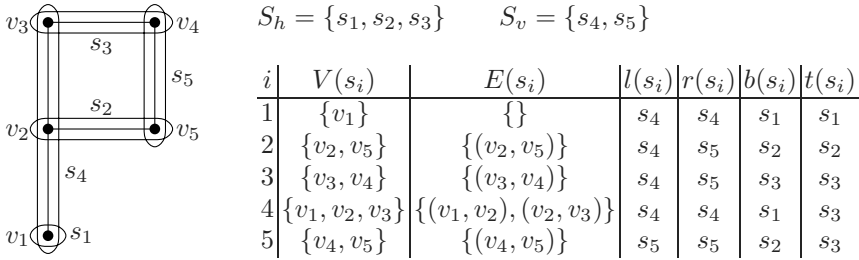


Fig. 4. Segments of a simple orthogonal grid drawing and its limits.

2.2 Shape Descriptions and Its Properties

Let $G = (V, E)$ be a graph with simple orthogonal representation H and segments $S_h \cup S_v$. A *shape description* of H is a tuple $\sigma = \langle D_h, D_v \rangle$ of so-called *constraint graphs*. Both graphs are directed and defined as $D_h = (S_v, A_h)$ and $D_v = (S_h, A_v)$. Thus, each node in D_h and D_v is a segment and the arc sets are given by

$$A_h = \{(l(e), r(e)) \mid e \in E_h\} \quad \text{and} \quad A_v = \{(b(e), t(e)) \mid e \in E_v\} .$$

The two digraphs characterize known relationships between the segments that must hold in any drawing of the graph because of its shape properties. Let $a = (s_i, s_j)$ be an arc in $A_h \cup A_v$. If $a \in A_v$ then the horizontal segment s_i must be placed at least one grid unit below segment s_j . For vertical segments the arc $a \in A_h$ expresses the fact that s_i must be to the left of s_j . Each arc is defined by at least one edge in E . Clearly, each vertical edge determines the relative position of two horizontal segments and vice versa. Figure 5 illustrates the shape description of a simple orthogonal grid drawing.

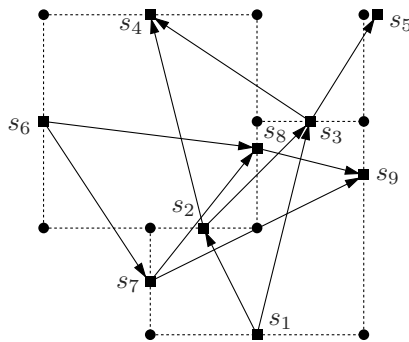


Fig. 5. A simple orthogonal drawing (dotted) and its shape description.

For two vertices v and w we use the notation $v \xrightarrow{*} w$ if there is a path from v to w . Shape descriptions have the following property:

Lemma 2. *Let $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ be a shape description. For every subsegment $s \in \mathcal{S}_h \cup \mathcal{S}_v$ the paths $l(s) \xrightarrow{*} r(s)$ and $b(s) \xrightarrow{*} t(s)$ are contained in $A_h \cup A_v$.*

Proof. We prove the lemma for horizontal subsegments. The proof for vertical subsegments is similar. By definition, the lemma holds for edges. Let s be a horizontal subsegment consisting of k consecutive edges $e_1, \dots, e_k$. With $l(e_1) = l(s)$, $r(e_k) = r(s)$ and $(l(e_i), r(e_i)) \in A_h$ and $b(s) = t(s) = \text{seg}(s)$ the result follows. $\square$

Definition 2. *Let $(s_i, s_j) \in \mathcal{S} \times \mathcal{S}$ be a pair of subsegments. We call the pair separated if and only if one of the following conditions holds:*

1. $r(s_i) \xrightarrow{*} l(s_j)$	3. $t(s_j) \xrightarrow{*} b(s_i)$
2. $r(s_j) \xrightarrow{*} l(s_i)$	4. $t(s_i) \xrightarrow{*} b(s_j)$

In an orthogonal drawing at least one of the four conditions must be satisfied for any such pair (s_i, s_j) . The following two observations show that we only need to consider separated segments of opposite direction (proofs omitted).

Observation 1. *Let $(s_i, s_j) \in \mathcal{S} \times \mathcal{S}$ be a pair of subsegments. If $(\text{seg}(s_i), \text{seg}(s_j))$ is separated then (s_i, s_j) is separated.*

Observation 2. *Assume that the arcs between the segments form an acyclic graph. Then the following is true: All segment pairs are separated if and only if all segment pairs of opposite directions are separated.*

The following lemma shows that we can restrict our focus to separated segments that share a common face. For a face f , we write $S(f)$ for the segments containing the horizontal and vertical edges on the boundary of f .

Lemma 3. *All segment pairs are separated if and only if for every face f the segment pairs $(s_i, s_j) \in S(f) \times S(f)$ are separated.*

Proof (Sketch). The forward direction follows immediately because of $S(f) \subseteq S$ for every face f . To show the other direction observe that the relative position of each pair of segments $(s_i, s_j) \in S(f) \times S(f)$ is fixed for any face f . Thus in a drawing respecting the placement of the underlying constraint graphs there are no crossings between segments of a face. Given that property we can show that there are no crossings at all in the drawing. It follows the separation of all segment pairs in the underlying constraint graphs. $\square$

2.3 Complete Extensions of Shape Descriptions

We will see next that any shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ can be extended so that the resulting constraint graphs correspond to a feasible orthogonal planar drawing. We give a characterization of these *complete extensions* in terms of properties of their constraint graphs.

Definition 3. A complete extension of a shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ is a tuple $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ with the following properties:

1. $A_h \subseteq B_h, A_v \subseteq B_v$.
2. B_h and B_v are acyclic.
3. Every segment pair is separated.

The following theorem characterizes the set of feasible solutions for the compaction problem.

Theorem 1. For any simple orthogonal drawing with shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ there exists a complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ of σ and vice versa: Any complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ of a shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ corresponds to a simple orthogonal drawing with shape description σ .

Proof. To prove the first part of the theorem, we consider a simple orthogonal grid drawing Γ with shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$. Let $c(s_i)$ denote the fixed coordinate for segment $s_i \in S_h \cup S_v$. We construct a complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ for σ as follows: $B_h = \{(s_i, s_j) \in S_v \times S_v \mid c(s_i) < c(s_j)\}$, i.e., we insert an arc from every vertical segment to each vertical segment lying to the right of s_i . Similarly, we construct the set B_v . Clearly, we have $A_h \subseteq B_h$ and $A_v \subseteq B_v$. We show the completeness by contradiction: Assume first that there is some pair (s_i, s_j) which is not separated. According to the construction this is only possible if the segments cross in Γ , which is a contradiction. Now assume that there is a cycle in one of the arc sets. Again, the construction of B_h and B_v forbids this case. Hence τ is a complete extension of σ .

We give a constructive proof for the second part of the theorem by specifying a simple orthogonal grid drawing for the complete extension τ . To accomplish this task we need to assign lengths to the segments. A *length assignment* for a complete extension of a shape description $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ is a function $c : S_v \cup S_h \rightarrow \mathbb{N}$ with the property $(s_i, s_j) \in B_h \cup B_v \Rightarrow c(s_i) < c(s_j)$. Given τ , such a function can be computed using any topological sorting algorithm in the acyclic graphs in τ , e.g., longest paths or maximum flow algorithms in the dual graph. For a fixed length assignment, the following simple and straightforward method assigns coordinates to the vertices. Let $x \in \mathbb{N}^V$ and $y \in \mathbb{N}^V$ be the coordinate vectors. Then simply setting $x_v = c(\text{vert}(v))$ and $y_v = c(\text{hor}(v))$ for every vertex $v \in V$ results in a correct grid drawing. The following points have to be verified:

1. All edges have positive integer length.

The length of a horizontal edge $e \in E_h$ is given by $c(r(e)) - c(l(e))$. We know that both values are integer and according to Lemma 2 that $(l(e), r(e)) \in B_h$ and thus $c(r(e)) > c(l(e))$. A similar argument applies to vertical edges.

2. Γ maps each circuit in G into a rectilinear polygon.

For a face f let $\Gamma(f)$ be the geometric representation of vertices and edges belonging to f . It is sufficient to show that $\Gamma(f)$ is a rectilinear polygon for each face f in G . Every vertex v on the boundary of f is placed according to the segments $\text{hor}(v)$ and $\text{vert}(v)$. Two consecutive vertices v and w on the boundary of f either share the same horizontal or the same vertical segment (since they are linked by an edge). Thus, either $x_v = x_w$ or $y_v = y_w$.

3. No subsegments cross.

Otherwise, assume that there are two such segments s_i and s_j that cross. Then $c(r(s_i)) \geq c(l(s_j))$, $c(r(s_j)) \geq c(l(s_i))$, $c(t(s_j)) \geq c(b(s_i))$ and $c(t(s_i)) \geq c(b(s_j))$ (see also Fig. 6). This is a contradiction to the completeness of τ . $\square$

We have transformed the compaction problem into a graph-theoretical problem. Our new task is to find a complete extension of a given shape description σ that minimizes the total edge length. If a shape description already satisfies the conditions of a complete extension (see Fig. 7(a)), the compaction problem can be solved optimally in polynomial time: The resulting problem is the dual of a maximum flow problem (see also Observation 3). Sometimes the shape description is not complete but it is only possible to extend it in one way (see Fig. 7(b)). In these cases also it is easy to solve the compaction problem. But in most cases it is not clear how to extend the shape description since there are many different possibilities (see Fig. 7(c)).

Observe that the preprocessing phase of our algorithm described at the beginning of Sect. 4 tests in polynomial time if there is a unique completion. In that case, our algorithm computes an optimal drawing in polynomial time.

3 An ILP Formulation for the Compaction Problem

The characterization given in the previous section can be used to obtain an integer linear programming formulation for the compaction problem COD. Let Γ be a simple orthogonal grid drawing of a graph $G = (V, E_h \cup E_v)$ with

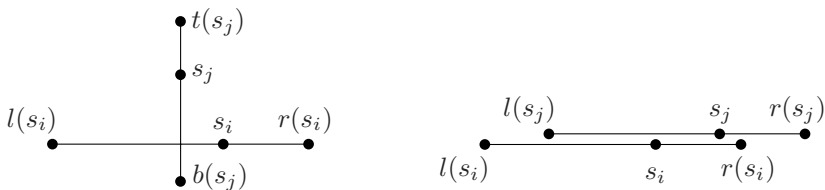


Fig. 6. Crossing subsegments of opposite (left) and same (right) direction.

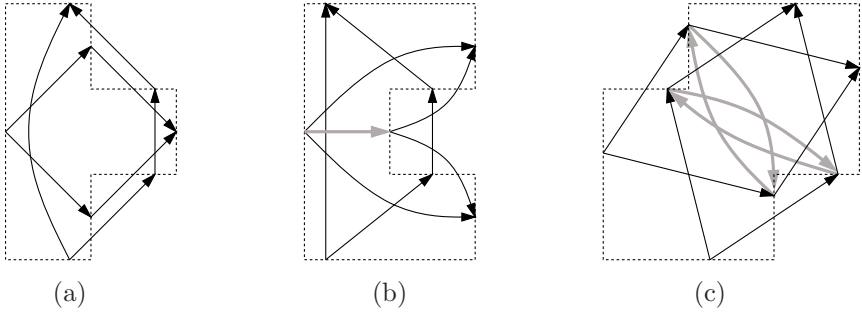


Fig. 7. Three types of shape descriptions. Dotted lines show the orthogonal grid drawings, thin arrows arcs in shape descriptions and thick gray arcs possible completions.

orthogonal representation H and let $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ be the corresponding shape description. The set of feasible solutions for this instance of COD can now be written as $\mathcal{C}(\sigma) = \{\tau \mid \tau \text{ is a complete extension of } \sigma\}$. Let $C = S_h \times S_h \cup S_v \times S_v$ be the set of possible arcs in the digraphs of σ . $\mathbb{Q}^{|C|}$ is the vector space whose elements are indexed with numbers corresponding to the members of C . For a complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ of σ we define an element $x^\tau \in \mathbb{Q}^{|C|}$ in the following way: $x_{ij}^\tau = 1$ if $(s_i, s_j) \in B_h \cup B_v$ and $x_{ij}^\tau = 0$ otherwise. We use these vectors to characterize the *Compaction Polytope* $P_{\text{COD}} = \text{conv}\{x^\tau \in \mathbb{Q}^{|C|} \mid \tau \in \mathcal{C}(\sigma)\}$.

In order to determine the minimum total edge length over all feasible points in P_{COD} we introduce a vector $c \in \mathbb{Q}^{|S_h \cup S_v|}$ to code the length assignment and give an integer linear programming formulation for the compaction problem COD. Let M be a very large number (for the choice of M see Lemma 5). The ILP for the compaction problem is the following:

$$\begin{aligned}
 (\text{ILP}) \quad & \min \sum_{e \in E_h} c_{r(e)} - c_{l(e)} + \sum_{e \in E_v} c_{t(e)} - c_{b(e)} \quad \text{subject to} \\
 & x_{ij} = 1 \quad \forall (s_i, s_j) \in A_h \cup A_v \quad (1.1) \\
 & x_{r(i),l(j)} + x_{r(j),l(i)} + x_{t(j),b(i)} + x_{t(i),b(j)} \geq 1 \quad \forall (s_i, s_j) \in S \times S \quad (1.2) \\
 & c_i - c_j + (M+1)x_{ij} \leq M \quad \forall (s_i, s_j) \in C \quad (1.3) \\
 & x_{ij} \in \{0, 1\} \quad \forall (s_i, s_j) \in C \quad (1.4)
 \end{aligned}$$

The objective function expresses the total edge length in a drawing for G . Note that the formulation also captures the related problem of minimizing the length of the longest edge in a drawing. In this case, the constraints $c_{r(e)} - c_{l(e)} \leq l_{\max}$ or $c_{t(e)} - c_{b(e)} \leq l_{\max}$ must be added for each edge e and the objective function must be substituted by $\min l_{\max}$. Furthermore, it is possible to give

each edge an individual weight in the objective function. In this manner, edges with higher values are considered more important and will preferably be assigned a shorter length.

We first give an informal motivation of the three different types of constraints and then show that any feasible solution of the ILP formulation indeed corresponds to an orthogonal grid drawing.

- (1.1) *Shape constraints.* We are looking for an extension of shape description σ . Since any extension must contain the arc sets of σ , the appropriate entries of x must be set to 1.
- (1.2) *Completeness constraints.* This set of constraints guarantees completeness of the extension. The respective inequalities model Definition 3.3.
- (1.3) *Consistency constraints.* The vector c corresponds to the length assignment and thus must fulfill the property $(s_i, s_j) \in B_h \cup B_v \Rightarrow c(s_i) < c(s_j)$. If $x_{ij} = 1$, then the inequality reads $c_j - c_i \geq 1$, realizing the property for the arc (i, j) . The case $x_{ij} = 0$ yields $c_i - c_j \leq M$ which is true if M is set to the maximum of the width and the height of Γ .

The following observation shows that we do not need to require integrality for the variable vector c . The subsequent lemma motivates the fact that no additional constraints forbidding the cycles are necessary.

Observation 3. *Let (x, c_f) with $x \in \{0, 1\}^{|C|}$ and $c_f \in \mathbb{Q}^{|S_h \cup S_v|}$ be a feasible solution for (ILP) and let z_f be the value of the objective function. Then there is a feasible solution (x, c) with $c \in \mathbb{N}^{|S_h \cup S_v|}$ and objective function value $z \leq z_f$.*

Proof. Since x is part of a feasible solution, its components must be either zero or one. Then (ILP) reads as the dual of a maximum flow problem with integer capacities. It follows that there is an optimal integer solution [2].

Lemma 4. *Let (x, c) be a feasible solution for (ILP) and let D_h and D_v be the digraphs corresponding to x . Then D_h and D_v are acyclic.*

Proof. Assume without loss of generality that there is a cycle $(s_{i_1}, s_{i_2}, \dots, s_{i_k})$ of length k in D_h . This implies $x_{i_1, i_2} = x_{i_2, i_3} = \dots = x_{i_k, i_1} = 1$ and the appropriate consistency constraints read

$$\begin{aligned} c_{i_1} - c_{i_2} + (M + 1) &\leq M \\ c_{i_2} - c_{i_3} + (M + 1) &\leq M \\ &\vdots \\ c_{i_k} - c_{i_1} + (M + 1) &\leq M \end{aligned}$$

Summed up, this yields $k \cdot (M + 1) \leq k \cdot M$, a contradiction. $\square$

Theorem 2. *For each feasible solution (x, c) of (ILP) for a shape description σ , there is a simple orthogonal grid drawing Γ whose shape corresponds to σ and vice versa. The total edge length of Γ is equal to the value of the objective function.*

Proof. For the first part of the proof, let x and c be the solution vectors of (ILP). According to Observation 3 we can assume that both vectors are integer. Vector x describes a complete extension τ of σ ; the extension property is guaranteed by constraints 1.1, completeness by constraints 1.2 and acyclicity according to Lemma 4. The consistency constraints 1.3 require c to be a length assignment for τ . The result follows with Theorem 1.

Again, we give a constructive proof for the second part: Theorem 1 allows us to use Γ for the construction of a complete extension τ for σ . Setting c_i to the fixed coordinate of segment s_i and x_{ij} to 1 if the arc (s_i, s_j) is contained in τ and to 0 otherwise, results in a feasible solution for the ILP. Evidently, the bounds and integrality requirements for c and x are not violated and Constraints 1.1 and 1.2 are clearly satisfied. To show that the consistency constraints hold, we consider two arbitrary vertical segments s_i and s_j . Two cases may occur: Assume first that s_i is to the left of s_j . Then the corresponding constraint reads $c_j - c_i \geq 1$ which is true since $c(j) > c(i)$ and the values are integer. Now suppose that s_i is not to the left of s_j . In this case the constraint becomes $c_i - c_j \leq M$ which is true for a sufficiently large M . A similar discussion applies to horizontal segments. Obviously, the value of the objective function is equal to the total edge length in both directions of the proof. $\square$

The integer linear program can be solved via a branch-and-cut or branch-and-bound algorithm. We now discuss the choice of the constant M .

Lemma 5. *The value $\max\{|S_h|, |S_v|\}$ is a sufficient choice for M .*

Proof. Note that any optimal drawing Γ has width $w \leq |S_v|$ and height $h \leq |S_h|$, otherwise a one-dimensional compaction step could be applied. M has to be big enough to “disable” Constraints 1.3 if the corresponding entry of x is zero; it has to be an upper bound for the distance between any pair of segments. Setting it to the size of the bigger of the two sets fulfills this requirement. $\square$

4 Implementation and Computational Results

4.1 Solving the ILP

Our implementation which we will refer to as OPT throughout this section, solves the integer linear program described in Sect. 3. It is realized as a compaction module inside the AGD-library [3, 1] and is written in C++ using LEDA [13, 12].

In a preprocessing phase, the given shape description σ is completed as far as possible. Starting with a list L of all segment pairs, we remove a pair (s_i, s_j) from L if it either fulfills the definition of separation or there is only one possibility to meet this requirement. In the latter case we insert the appropriate arc and proceed the iteration. The process stops if no such pair can be found in L . If the list is empty at the end of the preprocessing phase, we can solve the compaction problem in polynomial time by optimizing over the integral polyhedron given by the corresponding inequalities.

Otherwise, let σ^+ be the extension of σ resulting from the preprocessing phase. OPT first computes a corrupted layout for σ^+ in polynomial time. Then it checks which non-separated pairs in L induce indeed a violation of the drawing and adds the corresponding completeness and consistency inequalities. The resulting integer linear program is solved with the Mixed Integer Solver from CPLEX. The process of checking, adding inequalities and solving the ILP may have to be repeated because new pairs in L can cause a violation. Currently we are realizing a second implementation within the branch-and-cut framework ABACUS [7].

For the initial construction of L we exploit Lemma 3 and Observation 2. We only have to add pairs of segments which share a common face and which are of opposite direction. By decreasing the size of L in this way we could reduce the search space and speed up the computation remarkably.

4.2 Computational Results

We compare the results of our method to the results achieved by two other compaction methods: ORIG is an implementation of the original method proposed in [16]. It divides all the faces of the drawing into sub-faces of rectangular shape and assigns consistent edge lengths in linear time. 1DIM is an optimal one-dimensional compaction algorithm. It first calls ORIG to get an initial drawing and then runs iteratively a visibility-based compaction, alternating the direction in each run. It stops if no further one-dimensional improvement is possible.

The algorithms ORIG, 1DIM, and OPT have been tested on a large test set. This set, collected by the group of G. Di Battista, contains 11,582 graphs representing data from real-world applications [4]. For our experimental study, each of the graphs has been planarized by computing a planar subgraph and reinserting the edges, representing each crossing by a virtual vertex. After fixing the planar embedding for every graph we computed its shape using an extension of Tamassia's bend minimizing algorithm [10]. The resulting graphs with fixed shape served as input for the three compaction algorithms. We compared the total edge length and the area of the smallest enclosing rectangle of the drawings produced by ORIG, 1DIM, and OPT. Furthermore, we recorded their running times.

All the examples could be solved to optimality on a SUN Enterprise 10000. For all instances, the running times of ORIG and 1DIM have been below 0.05 and 0.43 seconds, respectively. OPT could solve the vast majority of instances (95.5%) in less than one second, few graphs needed more than five seconds (1.1%) and only 29 graphs needed more than one minute. The longest running time, however, was 68 minutes.

The average improvement of the total edge length computed by OPT over the whole test set of graphs was 2.4% compared to 1DIM and 21.0% compared to ORIG. Just looking at hard instances where OPT needed more than two seconds of running time we yield average improvements of 7.5% and 36.1%, resp. Figures 8 and 9 show this fact in more detail: The x -axis denotes the size of the graphs, the y -axis shows the improvement of total edge length in percent with respect to

1DIM and ORIG. We computed the minimal, maximal and average improvement for the graphs of the same size. The average improvement values are quite independent from the graph size, and the minimum and maximum values converge to them with increasing number of vertices. Note that OPT yields in some cases improvements of more than 30% in comparison to the previously best strategy, Fig. 1 in Sect. 1 shows the drawings for such a case (here, the improvement is 28%). For the area, the data look similar with the restriction that in few cases the values produced by 1DIM are slightly better than those from OPT; short edge length does not necessarily lead to low area consumption. The average area improvements compared to 1DIM and ORIG are 2.7% and 29.3%, however.

In general, we could make the following observations: Instances of the compaction problem COD divide into easy and hard problems, depending on the structure of the corresponding graphs. On the one hand, we are able to solve some randomly generated instances of biconnected planar graphs with 1,000 vertices in less than five seconds. In these cases, however, the improvement compared to the results computed by 1DIM is small. On the other hand, graphs containing tree-like structures have shown to be hard to compact since their number of fundamentally different drawings is in general very high. For these cases, however, the improvement is much higher.

5 Conclusions and Future Plans

We have introduced the constraint graphs describing the shape of a simple orthogonal grid drawing. Furthermore, we have established a direct connection between these graphs by defining *complete extensions* of the constraint graphs that satisfy certain connectivity requirements in both graphs. We have shown that complete extensions characterize the set of feasible drawings with the given shape. For a given complete extension we can solve the compaction problem COD in polynomial time. The graph-theoretical characterization allows us to formulate COD as an integer linear program. The preprocessing phase of our algorithm detects those instances having only one complete extension and, for them, the optimal algorithm runs in polynomial time. Our experiments show that the resulting ILP can be solved within short computation time for instances as big as 1,000 vertices.

There are still open questions concerning the two-dimensional compaction problem: We are not satisfied having the ‘big’ M in our integer linear programming formulation. Our future plans include the extensive study of the 0/1-polytope P_{COD} that characterizes the set of complete extensions. We hope that the results will lead to a running time improvement of our branch-and-cut algorithm. Independently, Bridgeman et al. [5] have developed an approach based on the upward planar properties of the layout graphs. We want to investigate a possible integration of their ideas. Furthermore, we plan to adapt our method to variations of the two-dimensional compaction problem.

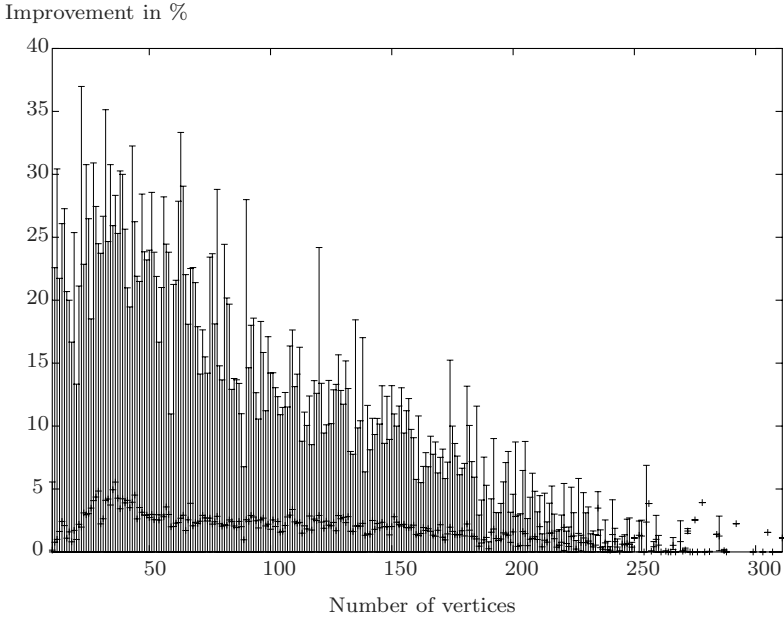


Fig. 8. Quality of OPT compared to 1DIM.

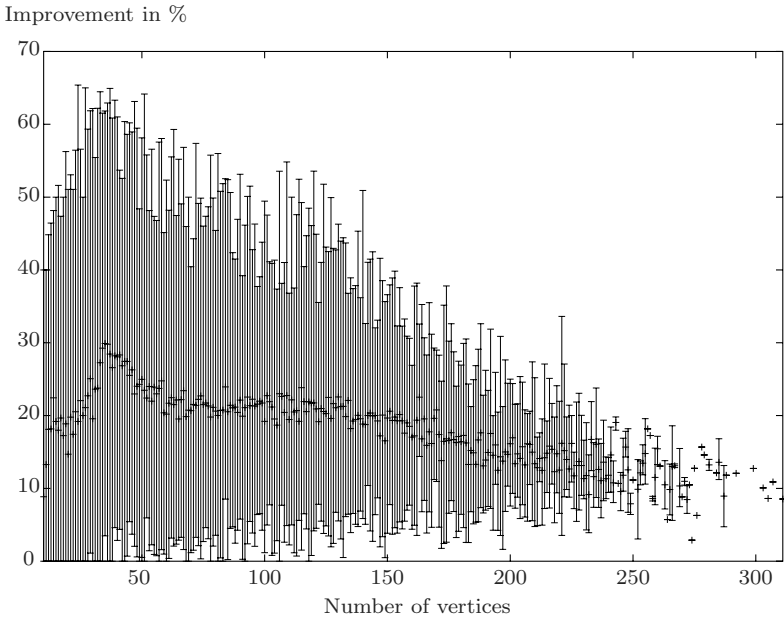


Fig. 9. Quality of OPT compared to ORIG.

References

- [1] AGD. *AGD User Manual*. Max-Planck-Institut Saarbrücken, Universität Halle, Universität Köln, 1998. <http://www.mpi-sb.mpg.de/AGD>.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.
- [3] D. Alberts, C. Gutwenger, P. Mutzel, and S. Näher. AGD-Library: A library of algorithms for graph drawing. In G. Italiano and S. Orlando, editors, *WAE '97 (Proc. on the Workshop on Algorithm Engineering)*, Venice, Italy, Sept. 11-13, 1997. <http://www.dsi.unive.it/~wae97>.
- [4] G. D. Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *CGTA: Computational Geometry: Theory and Applications*, 7:303 – 316, 1997.
- [5] S. Bridgeman, G. Di Battista, W. Didimo, G. Liotta, R. Tamassia, and L. Visмара. Turn-regularity and optimal area drawings of orthogonal representations. Technical report, Dipartimento di Informatica e Automazione, Università degli Studi di Roma Tre, 1999. To appear.
- [6] U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 254–266. Springer-Verlag, 1996.
- [7] M. Jünger and S. Thienel. Introduction to ABACUS - A Branch-And-CUt System. *Operations Research Letters*, 22:83–95, March 1998.
- [8] G. Kedem and H. Watanabe. Graph optimization techniques for IC-layout and compaction. *IEEE Transact. Comp.-Aided Design of Integrated Circuits and Systems*, CAD-3 (1):12–20, 1984.
- [9] G. W. Klau and P. Mutzel. Optimal compaction of orthogonal grid drawings. Technical Report MPI-I-98-1-031, Max-Planck-Institut für Informatik, Saarbrücken, December 1998.
- [10] G. W. Klau and P. Mutzel. Quasi-orthogonal drawing of planar graphs. Technical Report MPI-I-98-1-013, Max-Planck-Institut für Informatik, Saarbrücken, May 1998.
- [11] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, New York, 1990.
- [12] K. Mehlhorn, S. Näher, M. Seel, and C. Uhrig. LEDA Manual Version 3.7.1. Technical report, Max-Planck-Institut für Informatik, 1998. <http://www.mpi-sb.mpg.de/LEDA>.
- [13] K. Mehlhorn and S. Näher. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1):96–102, 1995.
- [14] M. Patrignani. On the complexity of orthogonal compaction. Technical Report RT-DIA-39-99, Dipartimento di Informatica e Automazione, Università degli Studi di Roma Tre, January 1999.
- [15] M. Schlag, Y.-Z. Liao, and C. K. Wong. An algorithm for optimal two-dimensional compaction of VLSI layouts. *Integration, the VLSI Journal*, 1:179–209, 1983.
- [16] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.

On the Number of Iterations for Dantzig-Wolfe Optimization and Packing-Covering Approximation Algorithms

Philip Klein^{1,*} and Neal Young^{2,**}

¹ Brown University, Providence, RI, USA; klein@cs.brown.edu

² Dartmouth College, Hanover, NH, USA; ney@cs.dartmouth.edu

1 Introduction

We start with definitions given by Plotkin, Shmoys, and Tardos [16]. Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and a polytope $P \subseteq \mathbb{R}^n$, the *fractional packing problem* is to find an $x \in P$ such that $Ax \leq b$ if such an x exists. An ϵ -*approximate* solution to this problem is an $x \in P$ such that $Ax \leq (1 + \epsilon)b$. An ϵ -*relaxed decision procedure* always finds an ϵ -approximate solution if an exact solution exists.

A **Dantzig-Wolfe-type** algorithm for a fractional packing problem $x \in P$, $Ax \leq b$ is an algorithm that accesses P only by *queries* to P of the following form: “given a vector c , what is an $x \in P$ minimizing $c \cdot x$?”

There are Dantzig-Wolfe-type ϵ -relaxed decision procedures (e.g. [16]) that require $O(\rho\epsilon^{-2} \log m)$ queries to P , where ρ is the *width* of the problem instance, defined as follows:

$$\rho(A, P) = \max_{x \in P} \max_i A_i \cdot x / b_i$$

where A_i denotes the i^{th} row of A .

In this paper we give a natural probability distribution of fractional packing instances such that, for an instance chosen at random, with probability $1 - o(1)$ *any* Dantzig-Wolfe-type ϵ -relaxed procedure must make *at least* $\Omega(\rho\epsilon^{-2} \log m)$ queries to P . This lower bound matches the aforementioned upper bound, providing evidence that the unfortunate linear dependence of the running times of these algorithms on the width and on ϵ^{-2} is an inherent aspect of the Dantzig-Wolfe approach.

The specific probability distribution we study here is as follows. Given m and ρ , let A be a random $\{0, 1\}$ -matrix with m rows and $n = \sqrt{m}$ columns, where each entry of A has probability $1/\rho$ of being 1. Let P be the n -simplex, and let b be the m -vector whose every entry is some v , where v is as small as possible so that $Ax \leq b$ for some $x \in P$.

The class of Dantzig-Wolfe-type algorithms encompasses algorithms and algorithmic methods that have been actively studied since the 1950’s through the current time, including:

* Research supported by NSF Grant CCR-9700146.

** Research supported by NSF Career award CCR-9720664.

- an algorithm by Ford and Fulkerson for multicommodity flow [14],
- Dantzig-Wolfe decomposition (generalized linear programming) [4],
- Benders’ decomposition [3],
- the Lagrangean relaxation method developed by Held and Karp and applied to obtaining lower bounds for the traveling salesman problem [9, 10],
- the multicommodity flow approximation algorithms of Shahrokhi and Matula [18], of Klein et al. [13], and of Leighton et al. [15],
- the covering and packing approximation algorithms of Plotkin, Shmoys, and Tardos [16] and the approximation algorithms of Grigoriadis and Khachiyan [8] for block-angular convex programs, and many subsequent works (e.g. [20, 6]).

In a later section we discuss some of the history of the above algorithms and methods and how they relate to the fractional packing problem studied here.

To prove the lower bound we use a probabilistic, discrepancy-theory argument to characterize the values of random $m \times s$ zero-sum games when s is much smaller than m . From the point of view proposed in [20], where fractional packing algorithms are derived using randomized rounding (and in particular the Chernoff bound), the intuition for the lower bound here is that it comes from the fact that the Chernoff bound is essentially tight.

Some of the multicommodity flow algorithms, and subsequently the algorithms of Plotkin, Shmoys, Tardos and of Grigoriadis and Khachiyan, use a more general model than the one described above. This model assumes the polytope P is the cross-product $P = P^1 \times \cdots \times P^k$ of k polytopes. In this model, each iteration involves optimizing a linear function over one of the polytopes P^i . It is straightforward to extend our lower bound to this model by making A block-diagonal, thus forcing each subproblem to be independently solved. In this general case, the lower bound shows that the number of iterations must be $\Omega(\epsilon^{-2}(\sum_i \rho_i) \log m)$, where ρ_i is the width of P^i . This lower bound is also tight within a constant factor, as it matches the upper bounds of Plotkin, Shmoys, and Tardos.

Previous Lower Bounds. In 1977, Khachiyan [12] proved an $\Omega(\epsilon^{-1})$ lower bound on the number of iterations to achieve an error of ϵ .

In 1994, Grigoriadis and Khachiyan proved an $\Omega(m)$ lower bound on the number of iterations to achieve a relative error of $\epsilon = 1$. They did not consider the dependence of the number of iterations on ϵ for smaller values of ϵ .

Freund and Schapire [5], in an independent work in the context of learning theory, prove a lower bound on the net “regret” of any adaptive strategy for playing repeated zero-sum games against an adversary. This result is related to, but different from, the result proved here.

2 Proof of Main Result

For any m -row n -column matrix A , define the *value* of A (considered as a two-player zero-sum matrix game) to be

$$V(A) \doteq \min_x \max_{1 \leq i \leq m} A_i \cdot x$$

where A_i denotes the i th row of A and x ranges over the n -vectors with non-negative entries summing to 1.

Theorem 1. *For $m \in \mathbb{N}$, $n = \Theta(m^{0.5})$, and $p \in (0, 1/2)$, let A be a random $\{0, 1\}$ $m \times n$ matrix with i.i.d. entries, each being 1 with probability p . Let $\bar{\epsilon} > 0$.*

With probability $1 - o(1)$,

- $V(A) = \Omega(p)$, and*
- for $s \leq \min\{(\ln m)/(p\bar{\epsilon}^2), m^{0.5-\delta}\}$ (where $\delta > 0$ is fixed), every $m \times s$ submatrix B of A satisfies*

$$V(B) > (1 + c\bar{\epsilon})V(A)$$

where c is a constant depending on δ .

Our main result follows as a corollary.

Corollary 1. *Let $m \in \mathbb{N}$, $\rho > 2$, and $\epsilon > 0$ be given such that $\rho\epsilon^{-2} = O(m^{0.5-\delta})$ for some constant $\delta > 0$.*

For $p = 1/\rho$, and $n = m^{0.5}$, let A be a random $\{0, 1\}$ $m \times n$ matrix as in the theorem. Let b denote the m -element vector whose every element is $V(A)$. Let $P = \{x \in \mathbb{R}^n : x \geq 0, \sum_i x_i = 1\}$ be the n -simplex.

Then with probability $1 - o(1)$, the fractional packing problem instance $x \in P, Ax \leq b$ has width $O(\rho)$, and any Dantzig-Wolfe-type ϵ -relaxed decision procedure requires at least $\Omega(\rho\epsilon^{-2} \log m)$ queries to P when given the instance as input.

Assuming the theorem for a moment, we prove the corollary. Suppose that the matrix A indeed has the two properties that hold with probability $1 - o(1)$ according to the theorem. It follows from the definition of $V(A)$ that there exists $x^* \in P^n$ such that $Ax^* \leq b$. That is, there exists a (non-approximate) solution to the fractional packing problem.

To bound the width, let $\bar{x}$ be any vector in P . By definition of P and A , for any row A_j of A we have $A_j \cdot \bar{x} \leq 1$. On the other hand, from the theorem we know that $V(A) = \Omega(p) = \Omega(1/\rho)$. Since $b_j = V(A)$, it follows that $A_j \cdot \bar{x}/b_j$ is $O(\rho)$. Since this is true for every j and $\bar{x} \in P$, this bounds the width.

Now consider any Dantzig-Wolfe-type ϵ -relaxed decision procedure. Suppose for a contradiction that it makes no more than $s \leq \rho(c\epsilon)^{-2} \ln m$ calls to the oracle that optimizes over P . In each of these calls, the oracle returns a vertex of P , i.e. a vector of the form

$$(0, 0, \dots, 0, 1, 0, \dots, 0, 0)$$

Let S be the set of vertices returned, and let $P(S)$ be the convex hull of these vertices. Every vector in $P(S)$ has at most s non-zero entries, for its only non-zero entries can occur in positions for which there is a vector in S having a 1 in that position. Hence, by the theorem with $\bar{\epsilon} = \epsilon/c$, there is no vector $x \in P(S)$ that satisfies $Ax \leq (1 + \epsilon)b$.

Consider running the same algorithm on the fractional packing problem $Ax \leq b, x \in P(S)$, i.e. with $P(S)$ replacing P . The procedure makes all the same queries to P as before, and receives all the same answers, and hence must give the same output, namely that an ϵ -approximate solution exists. This is an incorrect output, which contradicts the definition of a relaxed decision procedure.

3 Proof of Theorem 1

For any m -row n -column matrix A , define the *value* of A (considered as a two-player zero-sum matrix game) to be

$$V(A) = \min_x \max_{1 \leq i \leq m} A_i \cdot x$$

where A_i denotes the i th row of A and x ranges over the n -vectors with non-negative entries summing to 1.

Before we give the proof of Theorem 1, we introduce some simple tools for reasoning about $V(X)$ for a random $\{0, 1\}$ matrix X .

By the definition of V , $V(X)$ is at most the maximum, over all rows, of the average of the row's entries. Suppose each entry in X is 1 with probability q , and within any row of X the entries are independent. Then for any δ with $0 < \delta < 1$, a standard Chernoff bound implies that the probability that a given row's average exceeds $(1 + \delta)q$ is $\exp(-\Theta(\delta^2 q n_X))$, where n_X is the number of columns of X . Thus, by a naive union bound $\Pr[V(X) \geq (1 + \delta)q] \leq m_X \exp(-\Theta(\delta^2 q n_X))$ where m_X is the number of rows of X . For convenience we rewrite this bound as follows. For any $q \in [0, 1]$ and $\beta \in (0, 1]$, assuming $m_X/\beta \rightarrow \infty$,

$$\Pr[V(X) \geq (1 + \delta)q] = o(\beta) \text{ for some } \delta = O\left(\sqrt{\frac{\ln(m_X/\beta)}{q n_X}}\right). \quad (1)$$

We use an analogous *lower* bound on $V(X)$. By von Neumann's Min-Max Theorem

$$V(X) = \max_y \min_i X'_i \cdot y$$

(where X' denotes the transpose of X). Thus, reasoning similarly, if within any *column* of X (instead of any row) the entries are independent,

$$\Pr[V(X) \leq (1 - \delta)q] = o(\beta) \text{ for some } \delta = O\left(\sqrt{\frac{\ln(n_X/\beta)}{q m_X}}\right), \quad (2)$$

assuming $n_X/\beta \rightarrow \infty$. We will refer to (1) and (2) as the *naive upper and lower bounds* on $V(X)$, respectively.

Proof of Theorem 1.

The naive lower bound to $V(A)$ shows that

$$\Pr[V(A) \leq p(1 - \delta_0)] = o(1) \text{ for some } \delta_0 = O\left(\sqrt{\frac{\ln n}{pm}}\right) = o(1). \quad (3)$$

Thus, $V(A) \geq \Omega(p)$ with probability $1 - o(1)$.

Let $s = \min\{(\ln m)/(p\bar{\epsilon}^2), m^{0.5-\delta}\}$. Assume without generality that $s = (\ln m)/(p\bar{\epsilon}^2)$ (by increasing $\bar{\epsilon}$ if necessary).

We will show that with probability $1 - o(1)$ any $m \times s$ submatrix B of A has value

$$V(B) > (1 + c\bar{\epsilon})V(A) \quad (4)$$

The definition of value implies that $V(B') \geq V(B)$ for any $m \times s'$ submatrix B' of B (where $s' \leq s$). Thus we obtain (4) for such submatrices B' as well.

For any of the m rows of B , the expected value of the average of the s entries is p . We will show at least $r = s^2 \ln n$ of the rows have a higher than average number of ones and by focusing on these rows we will show that $V(B)$ is likely to be significantly higher than $V(A)$.

For appropriately chosen δ_1 , the probability that a given row of B has at least $\ell = (1 + \delta_1)ps$ ones is at least $\binom{s}{\ell} p^\ell (1-p)^{s-\ell} = \exp(-O(\delta_1^2 ps))$. (That is, the Chernoff bound is essentially tight here up to constant factors in the exponent.) Call any such row *good* and let G denote the number of good rows. In particular choosing some

$$\delta_1 = \Omega\left(\sqrt{\frac{\ln(m/r)}{ps}}\right) = \Omega\left(\sqrt{\frac{\ln m}{ps}}\right) = \Omega(\epsilon)$$

the probability that any given row is good is at least $2r/m$ and the expectation of G is at least $2r$. Since G is a sum of independent random $\{0, 1\}$ random variables, $\Pr[G < r] < \exp(-r/8)$.

By the choice of r , this is $o(1/n^s)$, so with probability $1 - o(1/n^s)$, B has at least r good rows.

Suppose this is indeed the case and select any r good rows. Let C be the $r \times s$ submatrix of B formed by the chosen rows. In any column of C , the entries are independent and by symmetry each has probability at least $p(1 + \delta_1)$ of being 1. Applying the naive lower bound (2) to $V(C)$, we find

$$\Pr[V(C) \leq p(1 + \delta_1)(1 - \delta_2)] = o(1/n^s) \text{ for some } \delta_2 = O\left(\sqrt{\frac{s \ln n}{pr}}\right). \quad (5)$$

By the choice of r , $\delta_2 = o(\delta_1)$. Thus $(1 + \delta_1)(1 - \delta_2) = 1 + \Omega(\delta_1)$. Since $V(B) \geq V(C)$, we find that, for any $m \times s$ submatrix B , $V(B) \geq p(1 + \Omega(\delta_1))$ with probability $1 - o(1/n^s)$.

Since there are at most $\binom{n}{s} \leq n^s$ distinct $m \times s$ submatrices B of A , the probability that all of them have value $p(1 + \Omega(\delta_1))$ is $1 - o(1)$. Finally, applying the naive upper bound to $V(A)$ shows that

$$\Pr[V(A) \geq p(1 + \delta_3)] = o(1) \text{ for some } \delta_3 = O\left(\sqrt{\frac{\ln m}{pn}}\right). \quad (6)$$

Since $\delta_3 = o(\delta_1)$, the result follows. $\square$

4 Historical Discussion

Historically, there are three lines of research within what we might call the Dantzig-Wolfe model. One line of work began with a method proposed by Ford and Fulkerson for computing multicommodity flow. Dantzig and Wolfe noticed that this method was not specific to multicommodity flow; they suggested decomposing an arbitrary linear program into two sets of constraints, writing it as

$$\min\{cx : x \geq 0, Ax \geq b, x \in P\} \quad (7)$$

and solving the linear program by an iterative procedure: each iteration involves optimizing over the polytope P . This approach, now called Dantzig-Wolfe decomposition, is especially useful when P can be written as a cross-product $P_1 \times \cdots \times P_k$, for in this case minimization over P can be accomplished by minimizing separately over each P_i . Often, for example, distinct P_i 's constrain disjoint subsets of variables. In practice, this method tends to require many iterations to obtain a solution with value optimum or nearly optimum, often too many to be useful.

Lagrangian Relaxation

A second line of research is represented by the work of Held and Karp [9, 10]. In 1970 they proposed a method for estimating the minimum cost of a traveling-salesman tour. Their method was based on the concept of a *1-tree*, which is a slight variant of a spanning tree. They proposed two ways to calculate this estimate; one involved formulating the estimate as the solution to the mathematical program

$$\max_u \left[ub + \min_{x \in P} (c - uA)x \right] \quad (8)$$

where P is the polytope whose vertices are the 1-trees. They suggested an iterative method to find an optimal or near-optimal solution: While they given some initial assignment to u , find a minimum-cost 1-tree with respect to the edge-costs $c - uA$. Next, update the node-prices u based on the degrees of the nodes in the 1-tree found. Find a min-cost 1-tree with respect to the modified costs, update the node-prices accordingly, and so on.

Like Dantzig and Wolfe's method, this method's only dependence on the polytope P is via repeatedly optimizing over it. In the case of Held and Karp's estimate, optimizing over P amounts to finding a minimum-cost spanning tree. Their method of obtaining an estimate for the solution to a discrete-optimization problem came to be known as Lagrangian relaxation, and has been applied to a variety of other problems.

Held and Karp's method for finding the optimal or near-optimal solution to (8) turns out to be the *subgradient method*, which dates back to the early sixties. Under certain conditions this method can be shown to converge in the limit, but, like Dantzig and Wolfe's method it can be rather slow. (One author refers to the "the correct combination of artistic expertise and luck" [19] needed to make progress in subgradient optimization.)

Fractional Packing and Covering

The third line of research, unlike the first two, provided guaranteed convergence rates. Shahrokhi and Matula [18] gave an approximation algorithm for a special case of multicommodity flow. Their algorithm was improved and generalized by Klein, Plotkin, Stein, and Tardos [13], Leighton et al. [15], and others. Plotkin, Shmoys, and Tardos [16] noticed that the technique could be generalized to apply to the problem of finding an element of the set

$$\{x : Ax \leq b, x \in P\} \quad (9)$$

where P is a convex set and A is a matrix such that $Ax \geq 0$ for every $x \in P$. In particular, as discussed in the introduction, they gave an **ϵ -relaxed decision procedure** that required $O(\rho\epsilon^{-2} \log m)$ queries to P , where ρ is the *width* of the problem instance.

A similar result was obtained independently by Grigoriadis and Khachiyan [8]. Many subsequent algorithms (e.g. [20, 6]) built on these results. Furthermore, many applications for these results have been proposed.

This method of Plotkin, Shmoys, Tardos and Grigoriadis, Khachiyan improves on Dantzig-Wolfe decomposition and subgradient optimization in that it does not require artistry to achieve convergence, and it is effective for reasonably large values of ϵ . However, for small ϵ the method is frustratingly slow. Might there be an algorithm in the Dantzig-Wolfe model that converges more quickly?

Our aim in this paper has been to address this question, and to provide evidence that the answer is no. However, our lower bound technique is incapable of proving a lower bound that is superlinear in m , the number of rows of A . The reason is that for any m -row matrix A , there is an m -column submatrix B such that $V(B) = V(A)$. This raises the question of whether there is a Dantzig-Wolfe-type method that requires a number of iterations polynomial in m but subquadratic in $1/\epsilon$.

References

- [1] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multicommodity flow. In *Proc. of the 34th IEEE Annual Symp. on Foundation of Computer Science*, pages 459–468, 1993.
- [2] B. Awerbuch and T. Leighton. Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks. In *Proc. of the 26th Ann. ACM Symp. on Theory of Computing*, pages 487–495, 1994.
- [3] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4:238–252, 1962.
- [4] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Res.*, 8:101–111, 1960.
- [5] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *J. Games and Economic Behavior*, to appear.
- [6] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proc. of the 39th Annual Symp. on Foundations of Computer Science*, pages 300–309, 1998.

- [7] A. V. Goldberg. A natural randomization strategy for multicommodity flow and related problems. *Information Processing Letters*, 42:249–256, 1992.
- [8] M. D. Grigoriadis and L. G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. Technical Report LCSR-TR-222, Rutgers University Computer Science Department, New Brunswick, NJ, April 1994.
- [9] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1971
- [10] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.
- [11] D. Karger and S. Plotkin. Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. In *Proc. of the 27th Ann. ACM Symp. on Theory of Computing*, pages 18–25, 1995.
- [12] L. G. Khachiyan. Convergence rate of the game processes for solving matrix games. *Zh. Vychisl. Mat. and Mat. Fiz.*, 17:1421–1431, 1977. English translation in *USSR Comput. Math and Math. Phys.*, 17:78–88, 1978.
- [13] P. Klein, S. Plotkin, C. Stein, and E. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, June 1994.
- [14] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multicommodity network flow. *Management Sci.*, 5:97–101, 1958.
- [15] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *Journal of Computer and System Sciences*, 50(2):228, 1995.
- [16] S. Plotkin, D. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.
- [17] T. Radzik. Fast deterministic approximation for the multicommodity flow problem. In *Proc. of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 486–492, 1995.
- [18] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *JACM*, 37:318–334, 1990.
- [19] J. F. Shapiro. A survey of lagrangean techniques for discrete optimization. *Annals of Discrete Mathematics*, 5:113–138, 1979.
- [20] N. E. Young. Randomized rounding without solving the linear program. In *Proc. of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 170–178, 1995.

Experimental Evaluation of Approximation Algorithms for Single-Source Unsplittable Flow

Stavros G. Kolliopoulos* and Clifford Stein**

¹ NEC Research Institute, Inc., 4 Independence Way, Princeton, NJ 08540
`stavros@research.nj.nec.com`

² Dartmouth College, Department of Computer Science, Hanover, NH 03755-3510
`cliff@cs.dartmouth.edu`

Abstract. In the *single-source unsplittable flow* problem, we are given a network G , a source vertex s and k commodities with sinks t_i and real-valued demands ρ_i , $1 \leq i \leq k$. We seek to route the demand ρ_i of each commodity i along a single s - t_i flow path, so that the total flow routed across any edge e is bounded by the edge capacity c_e . This NP-hard problem combines the difficulty of bin-packing with routing through an arbitrary graph and has many interesting and important variations. In this paper we initiate the experimental evaluation of approximation algorithms for unsplittable flow problems. We examine the quality of approximation achieved by several algorithms for finding a solution with near-optimal congestion. In the process we analyze theoretically a new algorithm and report on the practical relevance of heuristics based on minimum-cost flow. The experimental results demonstrate practical performance that is better than the theoretical guarantees for all algorithms tested. Moreover modifications to the algorithms to achieve better theoretical results translate to improvements in practice as well.

1 Introduction

In the *single-source unsplittable flow* problem (UFP), we are given a network $G = (V, E, u)$, a source vertex s and a set of k commodities with sinks $t_1, \dots, t_k$ and associated real-valued demands $\rho_1, \dots, \rho_k$. We seek to route the demand ρ_i of each commodity i , along a single s - t_i flow path, so that the total flow routed across any edge e is bounded by the edge capacity u_e . See Figure 1 for an example instance. The minimum edge capacity is assumed to have value at least $\max_i \rho_i$.

The requirement of routing each commodity on a single path bears resemblance to integral multicommodity flow and in particular to the multiple-source edge-disjoint path problem. For the latter problem, a large amount of work exists either for solving exactly interesting special cases (e.g. [10], [33], [32]) or for approximating, with limited success, various objective functions (e.g. [31], [11],

* Part of this work was performed at the Department of Computer Science, Dartmouth College while partially supported by NSF Award NSF Career Award CCR-9624828.

** Research partly supported by NSF Career Award CCR-9624828.

[21], [22],[19]). Shedding light on single-source unsplittable flow may lead to a better understanding of multiple-source disjoint-path problems. From a different perspective, UFP can be approached as a variant of standard, single-commodity, maximum flow and generalizes single-source edge-disjoint paths. UFP is also important as a unifying framework for a variety of scheduling and load balancing problems [20]. It can be used to model bin-packing [20] and certain scheduling problems on parallel machines [27]. Another application for UFP is in virtual-circuit routing, where the source represents a node wishing to multicast data to selected sinks. The conceptual difficulty in UFP arises from combining both packing constraints due to the existence of capacities and path selection through a graph of arbitrary topology.

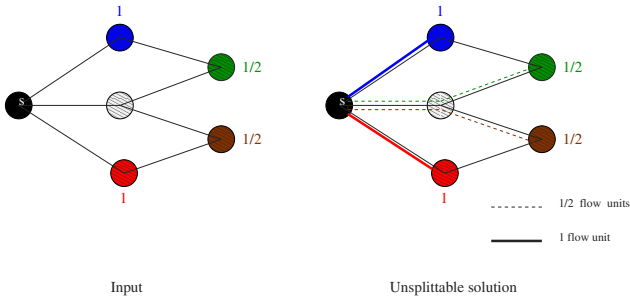


Fig. 1. Example UFP instance. Numbers on the vertices denote demands; all edge capacities are equal to 1.

The feasibility question for UFP, i.e., whether all commodities can be routed unsplittably, is strongly NP-complete [19]. Thus research has focused on efficient algorithms to obtain approximate solutions. For a minimization (resp. maximization) problem, a ρ -approximation algorithm, $\rho > 1$, ($\rho < 1$), outputs in polynomial time a solution with objective value at most (at least) ρ times the optimum. Different optimization versions can be defined such as minimizing congestion, maximizing the routable demand and minimizing the number of rounds [19]. In this work we focus on the congestion metric. The reader is referred to [20,24,7] for theoretical work on the other metrics.

Given a flow f define the *utilization* z_e of edge e as the ratio f_e/u_e . We seek an unsplittable flow f satisfying all demands, which minimizes *congestion*, i.e. the quantity $\max_e \{z_e, 1\}$. That is, the minimum congestion is the smallest number $\alpha \geq 1$, such that if we multiplied all capacities by α , f would satisfy the capacity constraints.

In this paper we initiate the experimental study of approximation algorithms for the unsplittable flow problem. Our study examines the quality of the effective approximation achieved by different algorithms, how this quality is affected by

input structure, and the effect of heuristics on approximation and running time. The experimental results indicate performance which is typically better than the theoretical guarantees. Moreover modifications to the algorithms to achieve better theoretical results translate to improvements in practice as well. The main focus of our work is to examine the quality of the approximation and we have not extensively optimized our codes for speed. Thus we have largely opted for maintaining the relative simplicity and modularity that we believe is one of the advantages of the algorithms we test. However keeping the running time low allows the testing of larger instances. Therefore we report on wall clock running times, at a granularity of seconds, as an indication of the time complexity in practice. Our codes run typically much faster than multicommodity flow codes on similar size instances [28,12,30] but worse than maximum flow codes [5] with which they have a comparable theoretical running time [24].

We chose to focus our experimentation on the congestion metric for a variety of reasons. The approximation ratios and integrality gaps known for congestion are in general better than those known for the other metrics [19,24,7]. Moreover congestion is relevant for maximizing routed demand in the following sense: an unsplittable solution with congestion λ provides a guarantee that $1/\lambda$ fraction of each demand can be routed while respecting capacities. Finally, the congestion metric has been extensively studied theoretically in the setting of randomized rounding technique [31] and for its connections to multicommodity flow and cuts (e.g. [26,17,25,18]).

Known theoretical performance guarantees. Kleinberg [19,20] was the first to consider UFP as a distinct problem and to give (large) constant-factor approximation algorithms. The authors obtained PARTITION, a simple 4-approximation algorithm [24] and a 3.33 bound for a more elaborate version of PARTITION. Very recently, Dinitz, Garg, and Goemans [7] obtained an algorithm with a performance guarantee of $1 + \max_i \{\rho_i\} \leq 2$. This new algorithm matches the integrality gap known for the problem.

New algorithms and heuristics. We provide a new algorithm, called H_PARTITION, and prove that it is a 3-approximation algorithm. We outline the ideas behind our algorithms in Section 3. For both PARTITION and H_PARTITION we test several codes, each implementing different heuristics. All heuristics are designed so that they do not affect the theoretical performance guarantee of the underlying base algorithm (4 and 3 for PARTITION and H_PARTITION respectively). The heuristics fall into two major categories. The *cost heuristic* invokes minimum-cost flow subroutines when the original algorithm would invoke maximum flow to compute partial unsplittable solutions. The cost of an edge e is defined as an appropriate increasing function of the current congestion on e . The *sparsification heuristic* discards at an early stage of the execution the edges carrying zero amount of flow in the initial solution to the maximum flow relaxation. The motivation behind the sparsification heuristic is to decrease the running time and observe how different versions of the algorithms behave in practice since the

zero edges are not exploited in the theoretical analysis. Finally, we test codes which combine both the sparsification and the cost heuristic.

Test data. We test our codes on five main families of networks. For each family we have generated an extensive set of instances by varying appropriate parameters. We test mostly on synthetic data adapted from previous experimental work for multicommodity flow [28,12] and cuts [3] together with data from some new generators. The instances produced are designed to accommodate a variety of path and cut characteristics. Establishing a library of interesting data generators for disjoint-path and unsplittable flow problems is an important task in its own right given the practical significance of this class of problems. We believe our work makes a first contribution in this direction. We attempted to obtain access to real-world data from telecommunication and VLSI applications, but were unsuccessful due to their proprietary nature. However one of the input families we test (LA networks) is adapted from synthetic data, used in the restoration capacity study of Deng et al. [6], which was created based on actual traffic forecasts on realistic networks.

Experimental Results. In summary, the experimental results indicate, for both algorithms PARTITION and H_PARTITION, typical performance which is better than the theoretical guarantees. Theoretical improvements translate to practical improvements as well since algorithm H_PARTITION achieved significantly better approximations than algorithm PARTITION on the instances we examined. We observed that the sparsification heuristic can drastically reduce the running time without much adverse effect on congestion. Finally we show that running the cost heuristic can provide additional small improvement on congestion at the expense of an increase in running time.

2 Preliminaries

A *single-source unsplittable flow problem* (UFP) $(G, s, \mathcal{T})$ is defined on a capacitated network $G = (V, E, u)$ where a subset $\mathcal{T} = \{t_1, \dots, t_k\}$ of V is the set of *sinks*; every edge e has a capacity u_e ; each sink t_i has a demand ρ_i . The pair (t_i, ρ_i) defines the *commodity* i . To avoid cumbersome notation, we use occasionally the set $\mathcal{T}$ to denote the set of commodities. We assume in our definition that each vertex in V hosts at most one sink; the input network can always be transformed to fulfill this requirement. A *source* vertex $s \in V$ is specified. We denote by $\rho_{\max}, \rho_{\min}$ the maximum and minimum demand values for the problem of interest. For any $I \subseteq \mathcal{T}$, a *routing* is a set of s - t_{i_j} paths P_j , $1 \leq j \leq |I|$ used to route ρ_{i_j} amount of flow from s to t_{i_j} for each $t_{i_j} \in I$. Given a routing g , the flow g_e through edge e is equal to $\sum_{P_i \in g | P_i \ni e} \rho_i$. For clarity we will refer to a standard maximum flow as *fractional maximum flow*. We make two assumptions in our treatment of UFP. First, we assume that there is a feasible fractional flow solution. This assumption is introduced for simplicity, our approximation guarantees hold even without this assumption. The second assumption is a *balance*

assumption that states that the maximum demand is less than or equal to the minimum edge capacity, i.e., a demand can be routed on any edge. This assumption is partly justified by the splitting of data into small size packets in actual networks so that no commodity has arbitrarily large demand. Even without the balance assumption a constant factor approximation is possible [24]. We will refer to instances that satisfy the balanced assumption as *balanced*, and ones which violate it as *unbalanced*, or more precisely, ones in which the maximum demand is ρ times the minimum capacity as ρ -unbalanced. Unless otherwise stated, we use n, m to denote $|V|, |E|$ respectively.

The following theorem is an easy consequence of the well-known augmenting path algorithm ([9], [8]) and will be of use.

Theorem 1. *Let $(G = (V, E, u), s, \mathcal{T})$ be a UFP on an arbitrary network G with all demands ρ_i , $1 \leq i \leq k$, equal to the same value σ , and edge capacities u_e equal to integral multiples of σ for all $e \in E$. If there is a fractional flow of value f , then there is an unsplittable flow of value at least f . This unsplittable flow can be found in time $O(km)$ where k is the number of sinks in $\mathcal{T}$.*

In the same way a routing corresponds to an edge flow, an edge flow can be represented as a path flow. Our algorithms will make use of the well-known *flow decomposition theorem* [1]. Given problem $(G, s, \mathcal{T})$ let a flow solution f be represented as an edge flow, i.e. an assignment of flow values to edges. Then one can find, in $O(nm)$ time, a representation of f as a path and cycle flow, where the paths join s to sinks in $\mathcal{T}$. In the remainder of the paper, we will assume that the cycles are deleted from the output of a flow decomposition.

3 The Two Basic Algorithms

We present first briefly the 4-approximation algorithm PARTITION. Subsequently we present and analyze the 3-approximation algorithm H_PARTITION. Recall that in our theoretical exposition we assume that in an UFP demands lie in the interval $(0, 1]$ and capacities are at least 1. In particular let $1/D$ be the minimum demand value for real $D \geq 1$.

PARTITION works by partitioning the original UFP into subproblems, each containing only demands in a fixed range. We use an initial maximum-flow solution to allocate capacity to different subproblems. Capacities in the subproblems are arbitrary and may violate the balance assumption. The subproblems are separately solved by subroutine INTERVAL_ROUTING which finds low-congestion solutions; the partial solutions are then combined by superimposing the flow paths obtained in the respective routings on the original network G . The combination step will have a subadditive effect on the total congestion and thus a near-optimal global solution is achieved. Subroutine INTERVAL_ROUTING deals with a subproblem Π that has demands in the range $(1/2^i, 1/2^{i-1}]$ as follows. Let f be a feasible fractional solution to Π . Round all demands up to $1/2^{i-1}$ and call Π' the resulting UFP instance. To obtain a feasible fractional solution for Π' , it suffices to multiply the flow f_e and the capacity u_e on each edge e

by a factor of at most 2. Further round all capacities up to the closest multiple of $1/2^{i-1}$ by adding at most $1/2^{i-1}$. The new edge capacities are now at most $2u_e + 1/2^{i-1}$. By Theorem 1 an unsplittable routing for Π' can now be found in polynomial time. Algorithm PARTITION is described formally in Fig. 2. Observe that an edge e which is assigned capacity $u_e^i = 0$ at Step 2, may still be used by subroutine INTERVAL_ROUTING on G_i to route up to a_i amount of flow.

Step 1. Find a feasible fractional solution f .

Step 2. Define a partition of the $[1/D, 1]$ interval into $\xi = \lfloor \log D \rfloor + 1$ consecutive subintervals $(1/2^{\lfloor \log D \rfloor + 1}, 1/2^{\lfloor \log D \rfloor}]$, $(1/2^{\lfloor \log D \rfloor}, 1/2^{\lfloor \log D \rfloor - 1}]$, $\dots$, $(1/2, 1]$. Construct ξ copies of G where the set of sinks in G_i , $1 \leq i \leq \xi$, is the subset $\mathcal{T}_i$ of $\mathcal{T}$ with demands in the interval $(1/2^i, 1/2^{i-1}]$. Using flow decomposition determine for every edge e the amount u_e^i of u_e used by f to route flow to sinks in $\mathcal{T}_i$. Set the capacity of edge e in G_i to u_e^i .

Step 3. Invoke INTERVAL_ROUTING on each G_i , $1 \leq i \leq \xi$, to obtain an unsplittable routing g^i . Output routing g , the union of the path sets g^i , $1 \leq i \leq \xi$.

Fig. 2. Algorithm PARTITION($G = (V, E, u)$, $s, \mathcal{T}$).

Theorem 2. [24] *Given an UFP Π , Algorithm PARTITION finds a 4-approximation for congestion in polynomial time.*

The running time is dominated by the time to perform $O(\log n)$ maximum flow computations and to perform flow decomposition. The fastest maximum flow algorithms currently have running time $O(nm \log_{\frac{m}{n \log n}} n)$ [13] and

$O(\min(n^{2/3}, m^{1/2})m \log(\frac{n^2}{m}) \log U)$ [14] when the capacities are integers in the range $[1, \dots, U]$. See [23] for exact computations of running times. In our implementations, we use a maximum flow routine repeatedly in order to perform flow decomposition.

3.1 A 3-Approximation Algorithm

An improved approximation algorithm can be obtained by combining the partitioning idea above with a more careful treatment of the subproblems. At the first publication of our results [24] we achieved a $(3.33 + o(1))$ -approximation. Subsequently we improved our scheme to obtain a 3 ratio; prior to this improvement, a 2-approximation for congestion was independently obtained by Dinitz, Garg and Goemans [7]. See [23] for a detailed exposition.

We give first without proof an application of the INTERVAL_ROUTING subroutine.

Lemma 1. *Let $\Pi = (G, s, \mathcal{T})$, be a UFP in which all demands have value $1/2^x$ for $x \in \mathbb{N}$, and all capacities are multiples of $1/2^{x+1}$, and let f be a fractional flow solution such that the flow f_e through any edge is a multiple of $1/2^{x+1}$. We*

can find in polynomial time an unsplittable routing g where the flow g_e through an edge e is at most $f_e + 1/2^{x+1}$.

Algorithm PARTITION finds a routing for each subproblem by scaling up subproblem capacities to ensure they conform to the requirements of Lemma 1. The new algorithm operates in phases, during which the number of distinct paths used to route flow for a particular commodity is progressively decreased. Call *granularity* the minimum amount of flow routed along a single path to any commodity, which has not been routed yet unsplittably. Algorithm PARTITION starts with a fractional solution of arbitrary granularity and essentially “in parallel” for all subproblems, rounds this fractional solution to an unsplittable one. In the new algorithm, we delay the rounding process in the sense that we keep computing a fractional solution of successively increasing granularity. The granularity will always be a power of $1/2$ and this unit will be doubled after each iteration. Once the granularity surpasses $1/2^x$, for some x , we guarantee that all commodities with demands $1/2^x$ or less have already been routed unsplittably. Every iteration improves the granularity at the expense of increasing the congestion. The method has the advantage that by Lemma 1, a fractional solution of granularity $1/2^x$ requires only a $1/2^x$ additive offset to current capacities to find a new flow of granularity $1/2^{x-1}$. Therefore, if the algorithm starts with a fractional solution of granularity $1/2^\lambda$, for some potentially large λ , the total increase to an edge capacity from then on would be at most $\sum_{j=1}^{\lambda} 1/2^j < 1$. Expressing the granularity in powers of $1/2$ requires an initial rounding of the demand values; this rounding will force us to scale capacities by at most a factor of 2. We first provide an algorithm for the case in which demand values are powers of $1/2$. The algorithm for the general case will then follow easily.

Theorem 3. *Let $\Pi = (G = (V, E, u), s, T)$ be a UFP, in which all demand values are of the form $2^{-\nu}$ with $\nu \in N$. There is an algorithm, 2H-PARTITION, which obtains, in polynomial time, an unsplittable routing such that the flow through any edge e is at most $zu_e + \rho_{\max} - \rho_{\min}$ where z is the optimum congestion.*

Remark: We first obtained a relative $(1 + \rho_{\max} - \rho_{\min})$ -approximation for the case with demand values in $\{1/2, 1\}$ in [24]. Dinitz, Garg and Goemans [7] first extended that theorem to arbitrary powers of $1/2$. Their derivation uses a different approach.

Proof. We describe the algorithm 2H-PARTITION. The crux of the algorithm is a relaxed decision procedure that addresses the following question. Is there an unsplittable flow after scaling all the input capacities by x ? The procedure will either return no or will output a solution where the capacity of edge e is at most $xu_e + \rho_{\max} - \rho_{\min}$. Embedding the relaxed decision procedure in a binary search completes the algorithm. See [16] for background on relaxed decision procedures. The following claim will be used to show the correctness of the procedure.

Claim. Given an UFP Π with capacity function r let Π' be the modified version of Π in which every edge capacity r_e has been rounded down to the closest

multiple r'_e of $1/2^\lambda$. If the optimum congestion is 1 for Π , the optimum congestion is 1 for Π' as well.

Proof of Claim. Let G be the network with capacity function r and G' be the network with rounded capacities r' . The amount $r_e - r'_e$ of capacity will be unused by any optimal unsplittable flow in G . The reason is that the sum $\sum_{i \in S} 1/2^i$, for any finite multiset $S \subset N$ is equal to a multiple of $1/2^{i_0}$, $i_0 = \min_{i \in S} \{i\}$. ■

We describe now the relaxed decision procedure. Let u' denote the scaled capacity function xu . Let $\rho_{\min} = 1/2^\lambda$. The relaxed decision procedure has at most $\lambda - \log(\rho_{\max}^{-1}) + 1$ iterations. During the first iteration, split every commodity in $\mathcal{T}$, called an *original commodity*, with demand $1/2^x$, $0 \leq x \leq \lambda$, into $2^{\lambda-x}$ *virtual commodities* each with demand $1/2^\lambda$. Call $\mathcal{T}_1$ the resulting set of commodities. Round down all capacities to the closest multiple of $1/2^\lambda$. By Theorem 1 we can find a fractional flow solution $f^0 \equiv g^1$, for $\mathcal{T}_1$, in which the flow through any edge is a multiple of $1/2^\lambda$. If this solution does not satisfy all demands the decision procedure returns no; by Claim 3.1 no unsplittable solution is possible without increasing the capacities u' . Otherwise the procedure continues as follows.

Observe that f^0 is a fractional solution for the set of original commodities as well, a solution with granularity $1/2^\lambda$. Let u_e^j denote the total flow through an edge after iteration j . The flow u_e^1 is trivially $f_e^0 \leq u_e^1 \leq u_e$. At iteration $i > 1$, we extract from g_{i-1} , using flow decomposition, the set of flow paths f^{i-1} used to route the set S_{i-1} of virtual commodities in $\mathcal{T}_{i-1}$ which correspond to original commodities in $\mathcal{T}$ with demand $1/2^{\lambda-i+1}$ or more. Pair up the virtual commodities in S_{i-1} corresponding to the same original commodity; this is possible since the demand of an original commodity is inductively an even multiple of the demands of the virtual commodities. Call the resulting set of virtual commodities $\mathcal{T}_i$. $\mathcal{T}_i$ contains virtual commodities with demand $1/2^{\lambda-i+1}$. The set of flow paths f^{i-1} is a fractional solution for the commodity set $\mathcal{T}_i$. By Lemma 1, we can find an unsplittable routing g^i for $\mathcal{T}_i$ such that the flow g_e^i through an edge e is at most $f_e^{i-1} + 1/2^{\lambda-i+2}$. Now the granularity has been doubled to $1/2^{\lambda-i+1}$. A crucial observation is that from this quantity, g_e^i , only an amount of at most $1/2^{\lambda-i+2}$ corresponds to new flow, added to e during iteration i . It is easy to see that the algorithm maintains the following two invariants.

INV1: After iteration i , all original commodities with demands $1/2^{\lambda-i+1}$ or less have been routed unsplittably.

INV2: After iteration $i > 1$, the total flow u_e^i though edge e , which is due to all i first iterations, satisfies the relation $u_e^i \leq u_e^{i-1} + 1/2^{\lambda-i+2}$.

Given that $u_e^1 \leq u'_e$ the total flow though e used to route unsplittably all commodities in $\mathcal{T}$ is at most

$$u'_e + \sum_{i=2}^{\lambda - \log(\rho_{\max}^{-1}) + 1} 1/2^{\lambda-i+2} = xu_e + \rho_{\max} - 1/2^\lambda = xu_e + \rho_{\max} - \rho_{\min}.$$

Therefore the relaxed decision procedure returns the claimed guarantee. For the running time, we observe that in each of the $l = \lambda - \log(\rho_{\max}^{-1}) + 1$ iterations,

it appears that the number of virtual commodities could be as large as $k2^l$, where k is the number of the original commodities. There are two approaches to ensure polynomiality of our method. One is to consider only demands of value $1/2^\lambda > 1/n^d$ for some $d > 1$. It is known [24] that we can route the very small demands while affecting the approximation ratio by an $o(1)$ factor. The other approach relies on the observation that at iteration i , the set S_{i-1} of flow paths yielded by the flow decomposition has size $O(km)$ since it corresponds to a fractional solution for a subset of the original commodities. In other words, virtual commodities in S_{i-1} corresponding to the same original commodity have potentially been routed along the same path in g^i . Hence, by appropriately merging virtual commodities corresponding to the same original commodity, and increasing the granularity at different rates for distinct original commodities, we can always run an iteration on a polynomial-size set of virtual commodities. We opted to present a “pseudopolynomial” version of the decision procedure for clarity. $\square$

Theorem 4. *Let $\Pi = (G = (V, E, u), s, \mathcal{T})$ be a UFP. There is an algorithm, H_PARTITION, which obtains in polynomial time a $\min\{3 - \rho_{\min}, 2 + 2\rho_{\max} - \rho_{\min}\}$ approximation for congestion.*

Proof. We describe the algorithm H_PARTITION. Round up every demand to its closest power of $1/2$. Call the resulting UFP Π' with sink set $\mathcal{T}'$. Multiply all capacities by at most 2. Then there is a fractional feasible solution f' to Π' . Let u' be the capacity function in Π' . The purpose of the rounding is to be able to express later on the granularity as a power of $1/2$. The remainder of the proof consists of finding an unsplittable solution to Π' ; this solution can be efficiently converted to an unsplittable solution to Π without further capacity increase. Such a solution to Π' can be found by the algorithm 2H_PARTITION from Theorem 3. Observe that the optimum congestion for Π' is at most z , the optimum congestion for Π . 2H_PARTITION will route through edge e flow that is at most

$$zu'_e + \rho'_{\max} - \rho'_{\min} \leq 2zu_e + \rho'_{\max} - \rho'_{\min},$$

where $\rho'_{\max}$ and $\rho'_{\min}$ denote the maximum and minimum demand values in Π' . But $\rho'_{\max} < 2\rho_{\max}$, $\rho'_{\max} \leq 1$, and $\rho'_{\min} \geq \rho_{\min}$ from the construction of Π' . Therefore $\rho'_{\max} - \rho'_{\min} \leq 1 - \rho_{\min}$ and $\rho'_{\max} - \rho'_{\min} \leq 2\rho_{\max} - \rho_{\min}$. Dividing the upper bound on the flow by $zu_e \geq 1$ yields the claimed guarantees on congestion. $\square$

4 Heuristics Used

The heuristics we designed fall into two major categories. We tested them both individually and in combination with each other.

The theoretical analysis of PARTITION and H_PARTITION is not affected if after finding the initial solution to the maximum flow relaxation, edges carrying

zero flow, called *zero edges* are discarded. The *sparsification heuristic* implements this observation. The motivation behind the sparsification heuristic is to decrease the running time and also observe how the theoretical versions of the algorithms behave in practice since the zero edges are not exploited in the theoretical analysis. On random graph instances where maximum flow solutions use typically a small fraction of the edges, the sparsification heuristic resulted in drastic reductions of the running time. We observed that the approximation guarantee typically degrades by a small amount when a large number of edges is discarded. However the reduction on running time and space requirements makes the sparsification heuristic attractive for large instances. We also tested an intermediate approach where a randomly chosen fraction of the zero edges (on the average between 30 and 50 percent) is retained.

The *cost heuristic* invokes a minimum-cost flow subroutine when the original algorithm would invoke maximum flow to compute partial unsplittable solutions. In algorithm PARTITION we use minimum cost flow at Step 3 to find the partial unsplittable solution to each subproblem. In algorithm H_PARTITION minimum-cost flow is used to compute at iteration i the unsplittable routing g_i for the set T_i of virtual commodities (cf. proof of Theorem 3). That is we use a minimum-cost flow routine to successively improve the granularity of the fractional solution. The cost of an edge e is defined as an appropriate increasing function of the current utilization of e . Let λ denote the current congestion. For robustness we use the ratio z_e/λ in the calculation of the cost $c(e)$. We tested using functions that depend either linearly or exponentially on z_e/λ . The objective was to discourage the repeated use of edges which were already heavily utilized. Exponential cost functions tend to penalize heavily the edge with the maximum utilization in a given set of edges. Their theoretical success in multicommodity flow work [34,18] motivated their use in our setting. In particular we let $c^L(e) = \beta_1 z_e/\lambda$ and $c^E(e) = 2^{\beta_2 z_e/\lambda}$ and tested both functions for different values of β_1, β_2 . Observe that an edge with utilization 0 so far will incur a cost of 1 under c^E , while 0 seems a more natural cost assignment for this case. However experiments suggested that the c^E function behaved marginally better than $c^E - 1$ by imposing limited moderation on the introduction of new edges in the solution.

5 Experimental Framework

Software and hardware resources. We conducted our experiments on a 400 MHz dual Pentium Pro PC with 500 MB of RAM and 130 MB swap space. None of the executions we report on here exceeded the system RAM or used the second system processor. The operating system was Linux, version 9. Our programs were written in C and compiled using gcc, version 2.7.2, with the -O3 optimization option.

Codes tested. For the implementation of the maximum flow and minimum cost flow subroutines of our algorithms we used the codes described in [5] and [4] respectively. We also used the Cherkassky-Goldberg code to implement the flow

decomposition step as described in Section 3. We assume integral capacities and demands in the unsplittable flow input. We test the following codes:

- h_prf**: this is algorithm PARTITION without sparsification.
- h_prf2**: this is algorithm PARTITION with intermediate sparsification. Zero cost edges are sampled with probability .3 and if the coin toss fails they are discarded.
- h_prf3**: this is algorithm PARTITION with all the zero edges discarded.
- 3a1**: this is algorithm H_PARTITION without sparsification. We implement a simplified version of algorithm H_PARTITION with theoretical guarantee 3 (in contrast to $3 - \rho_{\min}$ in Theorem 4).
- 3a12**: this is algorithm H_PARTITION with all the zero edges discarded.
- c/h_prf10y**: version of **h_prf** with the cost heuristic. Linear cost function with slope $\beta_1 = 10^y$.
- c/h_prfexpy**: version of **h_prf** with the cost heuristic. Exponential cost function with coefficient $\beta_2 = y$.
- c3/3a110y**: version of **3a1** with the cost heuristic. Linear cost function with slope $\beta_1 = 10^y$.
- c3/3a1expy**: version of **3a1** with the cost heuristic. Exponential cost function with coefficient $\beta_2 = y$.
- c3.2/3a1210y**: version of **3a12** with the cost heuristic. Linear cost function with slope $\beta_1 = 10^y$.
- c3.2/3a12expy**: version of **3a12** with the cost heuristic. Exponential cost function with coefficient $\beta_2 = y$.

Input classes. We generated data from four input classes. For each class we generated a variety of instances varying different parameters. The generators use randomness to produce different instances for the same parameter values. To make our experiments repeatable the seed of the pseudorandom generator is an input parameter for all generators. If no seed is given, a fixed default value is chosen. We used the default seed in generating all inputs. The four classes used follow.

genrmf. This is adapted from the GENRMF generator of Goldfarb and Grigoriadis [15,2]. The input parameters are **a b c1 c2 k d**. The generated network has b frames (grids) of size $a \times a$, for a total of $a * a * b$ vertices. In each frame each vertex is connected with its neighbors in all four directions. In addition, the vertices of a frame are connected one-to-one with the vertices of the next frame via a random permutation of those vertices. The source is the lower left vertex of the first frame. Vertices become sinks with probability $1/k$ and their demand is chosen uniformly at random from the interval $[1, d]$. The capacities are randomly chosen integers from $(c1, c2)$ in the case of interframe edges, and $(1, c2 * a * a)$ for the in-frame edges.

noigen. This is adapted from the noigen generator used in [3,29] for minimum-cut experimentation. The input parameters are **n d t p k**. The network has n nodes and an edge between a pair of vertices is included with probability

$1/d$. Vertices are randomly distributed among t components. Capacities are chosen uniformly from a prespecified range $[l, 2l]$ in the case of intercomponent edges and from $[pl, 2pl]$ for intracomponent edges, p being a positive integer. Only vertices belonging to one of the $t-1$ components not containing the source can become sinks, each with probability $1/k$. The desired effect of the construction is for commodities to contend for the light intercomponent cuts. Demand for commodities is chosen uniformly from the range $[1, 2l]$.

ranggen. This generates a random graph $G(n, p)$ with input parameters **n** **p** **c1** **c2** **k** **d**, where n is the number of nodes, p is the edge probability, capacities are in the range $(c1, c2)$, k is the number of commodities and demands are in the range $(0, d)$.

satgen. **satgen** first generates a random graph $G(n, p)$ as in **ranggen** and then uses the following procedure to designate commodities. Two vertices s and t are picked from G and maximum flow is computed from s to t . Let v be the value of the flow. New nodes corresponding to sinks are incrementally added each connected only to t and with a randomly chosen demand value. The process stops when the total demand reaches v , the value of the minimum s - t cut or when the number of added commodities reaches the input parameter k , typically given as a crude upper bound.

LA networks. This is a synthetic family of networks generated by Deng et al. [6] as part of the AT&T capacity restoration project. Although this project does not deal with unsplittable flow, the data readily gives rise to multiple-source unsplittable flow instances. The networks are derived from applying planarity-preserving edge contractions to a street map of Los Angeles. Pairs of terminals are placed on vertices following an exponential distribution arising from actual traffic forecasts. We transformed the instances to single-source by picking the largest degree vertex as the source.

program name	2		6		12		25		50		100	
	cong. time		cong. time		cong. time		cong. time		cong. time		cong. time	
3al	1.64	1	2.37	1	2.14	1	1.59	1	1.18	1	1.23	3
3al2	2.50	0	2.51	0	2.41	0	1.47	1	1.25	1	1.19	2
c/h_prf103	1.49	0	2.91	0	2.51	0	1.85	1	1.67	1	1.55	2
c/h_prfexp10	1.49	0	2.17	0	2.15	1	1.85	1	1.44	1	1.59	2
c3/3al103	2.56	0	2.29	1	1.96	2	1.46	2	1.25	4	1.16	7
c3/3alexp10	1.49	0	1.86	1	1.74	1	1.81	2	1.27	2	1.18	3
c3.2/3al2103	1.49	0	2.51	0	1.76	1	1.92	1	1.57	2	1.44	2
c3.2/3al2exp10	1.49	0	2.13	0	2.04	1	1.59	1	1.32	1	1.31	2
h_prf	2.00	0	2.47	0	2.51	0	2.15	0	1.50	0	1.45	1
h_prf2	2.00	0	2.47	0	2.51	0	2.14	0	1.50	1	1.41	2
h_prf3	2.00	0	2.75	0	2.20	0	1.74	0	1.36	0	1.46	1

Table 1. family noi_commodities: noigen 1000 1 2 10 i; 7981 edges; 2-unbalanced family; i is the expected percentage of sinks in the non-source component.

program name	2		4		8		16		32	
	cong. time		cong. time		cong. time		cong. time		cong. time	
3al	1.18	1	1.12	2	1.09	3	1.04	3	1.06	3
3al2	1.25	1	1.15	1	1.09	2	1.08	1	1.05	1
c/h_prf103	1.67	1	1.47	2	1.33	2	1.24	2	1.24	2
c/h_prfexp10	1.44	1	1.40	1	1.26	1	1.29	1	1.25	2
c3/3al103	1.25	4	1.22	6	1.13	7	1.12	5	1.07	6
c3/3alexp10	1.27	2	1.12	3	1.11	3	1.12	3	1.11	4
c3.2/3al2103	1.57	2	1.24	1	1.09	2	1.14	2	1.10	1
c3.2/3al2exp10	1.32	1	1.16	2	1.15	2	1.13	2	1.12	2
h_prf	1.50	1	1.42	1	1.30	1	1.22	1	1.24	2
h_prf2	1.55	1	1.40	1	1.22	2	1.23	2	1.18	2
h_prf3	1.36	0	1.22	1	1.38	0	1.28	1	1.20	1

Table 2. family noi_components: noigen 1000 1 i 10 50; 7981 edges; 2-unbalanced family; i is the number of components.

program name	32		64		128		256		512		1024	
	cong. time		cong. time		cong. time		cong. time		cong. time		cong. time	
3al	5.33	0	7.00	1	8.00	2	8.50	10	7.50	42	8.00	185
3al2	7.50	0	5.50	0	6.00	1	7.00	4	7.50	22	9.00	93
c/h_prf103	7.00	0	7.50	0	6.50	0	7.50	4	8.00	16	8.00	70
c/h_prfexp10	7.50	0	4.50	0	7.00	1	7.00	4	7.50	16	8.00	70
c3/3al103	8.00	0	4.50	1	8.00	2	7.50	12	7.50	63	7.50	278
c3/3alexp10	7.50	0	7.00	1	7.50	3	7.50	12	7.50	55	7.50	243
c3.2/3al2103	7.50	0	4.50	0	7.00	1	7.50	5	7.50	22	7.50	95
c3.2/3al2exp10	7.50	0	4.50	0	7.00	1	7.50	5	7.50	22	8.00	95
h_prf	7.50	0	7.00	0	6.00	1	9.50	3	7.50	16	8.50	70
h_prf2	7.50	0	7.00	0	7.50	1	10.00	4	10.50	16	8.50	70
h_prf3	5.33	0	7.00	0	9.50	0	7.50	0	7.50	3	8.00	11

Table 3. family ran_dense: rangen -a i*2 -b 30 -c1 1 -c2 16 -k i -d 16; 1228, 4890, 19733, 78454, 313922, 1257086 edges; 16-unbalanced family; i*2 is the number of vertices.

program name	2		5		10		20		50		70	
	cong. time		cong. time		cong. time		cong. time		cong. time		cong. time	
3al	1.88	7	1.91	22	1.65	44	1.35	94	1.16	259	1.14	372
3al2	1.74	4	1.97	13	1.55	24	1.31	49	1.18	143	1.14	201
c/h_prf103	2.41	6	2.49	13	2.10	27	1.81	63	1.66	178	1.65	234
c/h_prfexp10	2.67	6	2.43	11	2.09	22	1.79	53	1.74	149	1.71	201
c3/3al103	1.85	29	1.88	56	1.63	88	1.42	145	1.20	272	1.18	337
c3/3alexp10	1.80	21	1.96	30	1.64	42	1.46	70	1.24	175	1.18	235
c3.2/3al2103	1.80	5	1.97	18	1.68	31	1.47	58	1.26	150	1.20	207
c3.2/3al2exp10	1.92	8	2.00	20	1.67	31	1.37	54	1.21	144	1.22	196
h_prf	2.52	5	2.46	14	2.22	29	1.86	69	1.66	199	1.63	250
h_prf2	2.65	6	2.77	16	2.15	33	1.82	76	1.70	218	1.68	257
h_prf3	2.43	3	2.49	7	2.20	16	1.80	36	1.75	108	1.67	147

Table 4. family rmf_commDem: genrmf -a 10 -b 64 -c1 64 -c2 128 -k i -d 128; 29340 edges; 2-unbalanced family; i is the expected percentage of sinks in the vertices.

6 Experimental Results

In this section we give an overview of experimental results. We experimented on 13 different families and present data on several representative ones. Tables 1-6 provide detailed experimental results. The running time is given in seconds. The congestion achieved by algorithm PARTITION, on a balanced input, was typically less than 2.3 without any heuristics. The largest value observed was 2.89. Similarly for H_PARTITION the congestion, on a balanced input, was typically less than 1.65 and always at most 1.75. The theoretically better algorithm H_PARTITION does better in practice as well. To test the robustness of the approximation guarantee we relaxed, on several instances, the balance assumption allowing the maximum demand to be twice the minimum capacity or more, see Tables 1-5. The approximation guarantee was not significantly affected. Even in the extreme case when the balance assumption was violated by a factor of 16, as in Table 3, the best code achieved a congestion of 7.5. Performance on the smallest instances of the LA family (not shown here) was well within the typical values reported above. On larger LA instances, the exponential distribution generated an excessive number of commodities (on the order of 30,000 and up) for our codes to handle in the time frames we considered (a few minutes).

The sparsification heuristic proved beneficial in terms of efficiency as it typically cut the running time by half. One would expect that it would incur a degradation in congestion as it discards potentially useful capacity. This turned out not to be true in our tests. There is no clear winner in our experiments between `h_prf` and `h_prf3` or between `3a1` and `3a12`. One possible explanation is that in all cases the same maximum flow code is running in subroutines, which would tend to route flow in a similar fashion.

program name	2		4		16		64	
	cong.	time	cong.	time	cong.	time	cong.	time
3a1	2.26	0	2.18	0	1.48	9	1.21	214
3a12	1.82	0	2.72	0	1.46	4	1.20	102
c/h_prf103	2.45	0	3.10	0	1.88	5	1.68	140
c/h_prfexp10	2.45	0	3.01	0	1.98	4	1.73	117
c3/3a103	1.42	0	2.03	1	1.53	13	1.24	225
c3/3a1exp10	1.72	0	2.11	1	1.57	7	1.22	145
c3.2/3a12103	1.50	0	2.40	1	1.70	5	1.32	113
c3.2/3a12exp10	1.61	0	2.17	0	1.59	5	1.27	112
h_prf	1.85	0	3.56	0	2.05	6	1.72	154
h_prf2	1.85	0	3.56	0	1.94	6	1.75	171
h_prf3	2.38	0	3.12	0	1.92	3	1.72	87

Table 5. family `rmf_depthDem`: `genrmf -a 10 -b i -c1 64 -c2 128 -k 40 -d 128; 820, 1740, 7260, 29340 edges; 2-unbalanced family; i` is the number of frames.

In contrast, the cost heuristic achieved modest improvements with respect to the theoretical versions of the algorithms. The gains were more tangible when

the cost heuristic was applied to H_PARTITION which solves subproblems in an incremental fashion using information from the subproblems solved previously. On instances where 3a1 or 3a12 achieved a congestion of around 1.7 the gains were of the order of 5-10% depending on the actual cost function used. On instances where H_PARTITION did really well, achieving a congestion close to 1 the cost heuristic offered no improvement or even deteriorated the approximation.

Typical values of the cost function coefficients that achieved the best results were $\beta_1 = 10^3, 10^4$ for the linear case and $\beta_2 = 10$ or 25 for the exponential case. In this abstract we report results for $\beta_1 = 10^3$ and $\beta_2 = 10$. The exponential function performed slightly better than the linear function, with the gap closing as the network size or the number of commodities increased, in which case the actual numerical values for costs became less important.

program name	1		2		6		12		25		50	
	cong. time		cong. time		cong. time		cong. time		cong. time		cong. time	
3a1	1.55	1	1.56	1	1.67	5	1.64	11	1.64	27	1.70	64
3a12	1.45	0	1.67	1	1.78	3	1.57	6	1.67	15	1.75	35
c/h_prf103	1.80	1	2.11	0	2.22	2	2.15	4	2.33	11	2.33	26
c/h_prfexp10	1.92	0	2.11	1	2.11	2	2.15	5	2.33	11	2.33	24
c3/3a1103	1.44	1	1.64	2	1.44	7	1.67	14	1.56	30	1.56	130
c3/3a1exp10	1.30	1	1.33	2	1.56	7	1.57	14	1.56	34	1.56	78
c3.2/3a12103	1.15	1	1.40	1	1.44	3	1.56	6	1.56	15	1.56	34
c3.2/3a12exp10	1.27	0	1.44	1	1.44	3	1.56	6	1.56	14	1.57	34
h_prf	1.92	0	2.11	1	2.11	2	2.33	4	2.22	11	2.33	27
h_prf2	1.92	0	2.22	0	2.33	2	2.56	4	2.60	12	2.89	28
h_prf3	2.00	0	2.20	1	2.18	1	2.33	1	2.22	3	2.42	9

Table 6. family sat_density: satgen -a 1000 -b i -c1 8 -c2 16 -k 10000 -d 8; 9967, 20076, 59977, 120081, 250379, 500828 edges; 22, 61, 138, 281, 682, 1350 commodities; i is the expected percentage of pairs of vertices joined by an edge.

In summary, algorithm H_PARTITION achieved significantly better approximations than algorithm PARTITION on the instances we examined. The sparsification heuristic can reduce drastically the running time. Running the cost heuristic can provide additional improvement on congestion at the expense of an increase in running time.

7 Future Work

There are several directions to pursue in future work. The main task would be to implement the new 2-approximation algorithm of Dinitz et al. [7] and see how it behaves in practice. Testing on data with real network characteristics is important if such instances can be made available. Designing new generators that push the algorithms to meet their theoretical guarantees will enhance our understanding of the problem. Finally, a problem of minimizing makespan on

parallel machines [27] reduces to single-source unsplittable flow on a two-level graph with simple topology. It would be interesting to see how the unsplittable flow algorithms behave on actual instances of this problem.

Acknowledgement. Thanks to Aravind Srinivasan for useful discussions and for motivating this project. An early version of a $G(n, p)$ generator was written by Albert Lim. We also wish to thank Michel Goemans for sending us a copy of [7]. We are grateful to Jeff Westbrook for giving us access to the LA networks. The first author thanks Peter Blicher and Satish Rao for useful discussions.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.
2. Tamas Badics. GENRMF. <ftp://dimacs.rutgers.edu/pub/netflow/generators/-network/genrmf/>, 1991.
3. C. Chekuri, A. V. Goldberg, D. Karger, M. Levine, and C. Stein. Experimental study of minimum cut algorithms. In *Proc. of 8th SODA*, 324–333, 1997.
4. B. V. Cherkassky and A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1997.
5. B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.
6. M. Deng, D. F. Lynch, S. J. Phillips, and J. R. Westbrook. Algorithms for restoration planning in a telecommunications network. In *ALENEX99*, 1999.
7. Y. Dinitz, N. Garg, and M. Goemans. On the single-source unsplittable flow problem. In *Proc. of 39th FOCS*, 290–299, November 1998.
8. P. Elias, A. Feinstein, and C. E. Shannon. Note on maximum flow through a network. *IRE Transactions on Information Theory IT-2*, 117–199, 1956.
9. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canad. J. Math.*, 8:399–404, 1956.
10. A. Frank. Packing paths, cuts and circuits - a survey. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*, 49–100. Springer-Verlag, Berlin, 1990.
11. N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.
12. A. V. Goldberg, J. D. Oldham, S. A. Plotkin, and C. Stein. An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow. In *Proc. of the 6th Conference on Integer Programming and Combinatorial Optimization*, 338–352, 1998. Published as Lecture Notes in Computer Science 1412, Springer-Verlag”.
13. A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.
14. A.V. Goldberg and S. Rao. Beyond the flow decomposition barrier. In *Proc. of the 38th Annual Symposium on Foundations of Computer Science*, 2–11, 1997.
15. D. Goldfarb and M. Grigoriadis. A computational comparison of the Dinic and Network Simplex methods for maximum flow. *Annals of Operations Research*, 13:83–123, 1988.
16. D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

17. P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. In *Proc. of the 31st Annual Symposium on Foundations of Computer Science*, 726–737, 1990.
18. P. Klein, S. A. Plotkin, C. Stein, and É. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, June 1994.
19. J. M. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, MIT, Cambridge, MA, May 1996.
20. J. M. Kleinberg. Single-source unsplittable flow. In *Proc. of the 37th Annual Symposium on Foundations of Computer Science*, 68–77, October 1996.
21. J. M. Kleinberg and É. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. In *Proc. of the 27th Annual ACM Symposium on Theory of Computing*, 26–35, 1995.
22. J. M. Kleinberg and É. Tardos. Disjoint paths in densely-embedded graphs. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science*, 52–61, 1995.
23. S. G. Kolliopoulos. *Exact and Approximation Algorithms for Network Flow and Disjoint-Path Problems*. PhD thesis, Dartmouth College, Hanover, NH, August 1998.
24. S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Proc. of the 38th Annual Symposium on Foundations of Computer Science*, 426–435, 1997.
25. T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *Journal of Computer and System Sciences*, 50:228–243, 1995.
26. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proc. of the 29th Annual Symposium on Foundations of Computer Science*, 422–431, 1988.
27. J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.
28. T. Leong, P. Shor, and C. Stein. Implementation of a combinatorial multicommodity flow algorithm. In D. S. Johnson and C. C. McGeoch, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: Volume 12, Network Flows and Matching*. October 1991.
29. Hiroshi Nagamochi, Tadashi Ono, and Toshihde Ibaraki. Implementing an efficient minimum capacity cut algorithm. *Mathematical Programming*, 67:325–241, 1994.
30. CPLEX Optimization, Inc. Using the CPLEX callable library. Software Manual, 1995.
31. P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
32. N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*. Springer-Verlag, Berlin, 1990.
33. A. Schrijver. Homotopic routing methods. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*. Springer-Verlag, Berlin, 1990.
34. F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37:318 – 334, 1990.

Approximation Algorithms for a Directed Network Design Problem

Vardges Melkonian* and Éva Tardos**

Cornell University, Ithaca, NY 14853
vardges@orie.cornell.edu, eva@cs.cornell.edu

Abstract. We present a 2-approximation algorithm for a class of *directed* network design problems. The network design problem is to find a minimum cost subgraph such that for each vertex set S there are at least $f(S)$ arcs leaving the set S . In the last 10 years general techniques have been developed for designing approximation algorithms for *undirected* network design problems. Recently, Kamal Jain gave a 2-approximation algorithm for the case when the function f is weakly supermodular. There has been very little progress made on *directed* network design problems. The main techniques used for the undirected problems do not extend to the directed case.

András Frank has shown that in a special case when the function f is intersecting supermodular the problem can be solved optimally. In this paper, we use this result to get a 2-approximation algorithm for a more general case when f is crossing supermodular. We also extend Jain's techniques to directed problems. We prove that if the function f is crossing supermodular, then any basic solution of the LP relaxation of our problem contains at least one variable with value greater or equal to $1/4$.

1 Introduction

We consider the following network design problem for directed networks. Given a directed graph with nonnegative costs on the arcs find a minimum cost subgraph where the number of arcs leaving set S is at least $f(S)$ for all subsets S . Formally, given a directed graph $G = (V, E)$ and a requirement function $f : 2^V \mapsto Z$, the network design problem is the following integer program:

$$\text{minimize } \sum_{e \in E} c_e x_e \tag{1}$$

subject to

$$\begin{aligned} \sum_{e \in \delta_G^+(S)} x_e &\geq f(S), & \text{for each } S \subseteq V, \\ x_e &\in \{0, 1\}, & \text{for each } e \in E, \end{aligned}$$

* Research partially supported by NSF grant CCR-9700163.

** Research partially supported by NSF grant CCR-9700163, and ONR grant N00014-98-1-0589.

where $\delta_G(S)$ denotes the set of arcs leaving S . For simplicity of notation we will use $x(\delta_G(S))$ to denote $\sum_{e \in \delta_G(S)} x_e$.

The following are some special cases of interest. When $f(S) = k$ for all $\emptyset \neq S \subset V$ the problem is that of finding a minimum cost k -connected subgraph. The directed Steiner tree problem is to find the minimum cost directed tree rooted at r that contains a subset of vertices $D \subseteq V$. This problem is a network design problem where $f(S) = 1$ if $r \notin S$ and $S \cap D \neq \emptyset$ and $f(S) = 0$ otherwise. Both of these special cases are known to be NP-complete. In fact, the directed Steiner tree problem contains the Set Cover problem as a special case, and hence no polynomial time algorithm can achieve an approximation better than $O(\log n)$ unless $P=NP$ [10].

In the last 10 years there has been significant progress in designing approximation algorithm for *undirected* network design problems [1,5,12,4,8], the analog of this problem where the graph G is undirected. General techniques have been developed for the undirected case, e.g., primal-dual algorithms [1,5,12,4]. Recently, Kamal Jain gave a 2-approximation algorithm for the *undirected* case when the function f is weakly supermodular. The algorithm is based on a new technique: Jain proved that in any basic solution to the linear programming relaxation of the problem *there is* a variable whose value is at least a half.

There has been very little progress made on *directed* network design problems. The main general technique used for approximating undirected network design problems, the primal-dual method does not extend to the directed case. Charikar et al. [2] gave the only nontrivial approximation algorithm for the directed Steiner tree problem. Their method provides an $O(n^\epsilon)$ -approximation algorithm for any fixed ϵ . Khuller and Vishkin [9] gave a simple 2-approximation algorithm for the special case of the k -connected subgraph problem in undirected graphs. A similar method also works for the directed case.

In this paper, we consider the case when f is crossing supermodular, i.e., for every $A, B \subseteq V$ such that $A \cap B \neq \emptyset$ and $A \cup B \neq V$ we have that

$$f(A) + f(B) \leq f(A \cap B) + f(A \cup B). \quad (2)$$

Note that the k -connected subgraph problem is defined by the function $f(S) = k$ for all $\emptyset \neq S \subset V$, which is crossing supermodular. Hence the network design problem with a crossing supermodular requirement function f is still NP-hard. However, the function defining the directed Steiner tree problem is not crossing supermodular.

The paper contains two main results: a 2-approximation algorithm for the integer program, and a description of the basic solutions of the LP-relaxation.

The network design problem with a crossing supermodular requirement function f is a natural extension of the case of intersecting supermodular function considered by Frank [11], i.e., when the inequality (2) holds whenever A and B are intersecting. Frank [11] has shown that this special case can be solved optimally. However, when f is crossing supermodular the problem is NP-hard and no approximation results are known for this general case. In this paper, we combine Frank's result and the k -connected subgraph approximation of Khuller and

Vishkin [9] to obtain a 2-approximation algorithm for the crossing supermodular case.

In the second part of the paper, we describe the basic solutions of the LP relaxation of our problem. Extending Jain's technique [8] to the case of directed graphs, we show that in any basic feasible solution of the LP relaxation (where we replace the $x_e \in \{0, 1\}$ constraints by $0 \leq x_e \leq 1$ for all arcs e), there is at least one variable with value at least quarter. Using this result, we can obtain a 4-approximation algorithm for the problem by iteratively rounding all variables above a quarter to 1.

An interesting special case of crossing supermodular function arises when the requirement of a subset S is merely a function of its cardinality $|S|$, i.e., $f(S) = g(|S|)$. It is easy to show that requiring f to be crossing supermodular is equivalent to requiring that g to be a convex function. This particularly includes the case when $f(S) = k$ for all sets. For this latter problem, the best known approximation factor is 2 [9]. Another example of the crossing supermodular function is $f(S) = \max(|S|, k)$. Note that sets S that contain almost all nodes have a very large value in this function. A more interesting function for the context of network design would be the function $f(S) = \min(|S|, k, |V \setminus S|)$. However, the minimum of convex functions is not convex, and this function is not crossing supermodular. It is a very interesting open problem if these techniques extend to the class of weakly supermodular functions, that were considered by Jain in the undirected case. Note that this more general case includes the Steiner tree problem, and hence we cannot hope for a constant approximation algorithm.

The rest of the paper is organized as follows. In the next section we give the 2-approximation algorithm. The main theorem stating that all basic solutions to the linear programming relaxation of (1) have a variable that is at least a quarter is given in Section 3. In Section 4 we sketch the 4-approximation algorithm that is based on this theorem. This part of the paper is analogous to Jain's paper [8] for the undirected case.

2 The 2-Approximation Algorithm

We consider the following network design problem:

$$\text{minimize } \sum_{e \in E} c_e x_e \quad (3)$$

subject to

$$\begin{aligned} \sum_{e \in \delta_G(S)} x_e &\geq f(S), & \text{for each } S \in \rho, \\ x_e &\in \{0, 1\}, & \text{for each } e \in E, \end{aligned}$$

where $\delta_G(S)$ denotes the set of the arcs leaving S , and $f(S)$ is a crossing supermodular function on the set system $\rho \subseteq 2^V$.

Frank [11] has considered the special case of this problem when f is intersecting supermodular. He showed that in this case, the LP relaxation is totally dual integral (TDI), and gave a direct combinatorial algorithm to solve the problem. The case of crossing supermodular function is NP-hard (as it contains the k -connected subgraph problem as a special case). Here we use Frank's theorem to get a simple 2-approximation algorithm.

Let r be a vertex of G . Divide all the sets in ρ into two groups as follows:

$$\rho_1 = \{A \in \rho : r \notin A\}; \quad (4)$$

$$\rho_2 = \{A \in \rho : r \in A\}. \quad (5)$$

The idea is that using Frank's technique we can solve the network design problem separately for ρ_1 and ρ_2 , and then combine the solutions to get a 2-approximation for the original problem.

Lemma 1. *For the set systems ρ_1 and ρ_2 , the problem (3) can be solved optimally.*

Proof. For any $S \in \rho_1$, define the requirement function as $f_1(S) = f(S)$. Then the problem for the first collection ρ_1 is

$$\text{minimize } \sum_{e \in E} c_e x_e \quad (6)$$

subject to

$$\begin{aligned} \sum_{e \in \delta_G(S)} x_e &\geq f_1(S), & \text{for each } S \in \rho_1, \\ x_e &\in \{0, 1\}, & \text{for each } e \in E, \end{aligned}$$

Note that f_1 is intersecting supermodular on the set family ρ_1 . For any $S_1, S_2 \in \rho_1$, we have $r \notin S_1 \cup S_2$, and so $S_1 \cup S_2 \neq V$. This together with crossing supermodularity of f implies that the requirement function $f_1(S)$ is intersecting supermodular on ρ_1 , and hence the LP relaxation of (6) is TDI, and the integer program can be solved optimally.

We will use a similar idea for ρ_2 . This is not an intersecting set system. To be able to apply Schrijver's theorem we will consider the reverse of the graph G defined as $G^r = (V, E^r)$ where $E^r = \{i \rightarrow j : j \rightarrow i \in E\}$. If $e = i \rightarrow j \in E$, then for $e^r = j \rightarrow i \in E^r$ define $c_{e^r} = c_e$.

Note that the network design problem with the function $f_2(S) = f(S)$ for $S \in \rho_2$ is equivalent to the problem on the reverse graph G^r with requirement function $f_2^r(V \setminus S) = f_2(S)$ for $S \in \rho_2$. Let $\rho_2^r = \{A \subseteq V : V \setminus A \in \rho_2\}$ and $f_2^r(S) = f(V \setminus S)$ for any $S \in \rho_2^r$. Then the second subproblem is

$$\text{minimize } \sum_{e \in E^r} c_e x_e \quad (7)$$

subject to

$$\begin{aligned} \sum_{e \in \delta_{G^r}(S)} x_e &\geq f_2^r(S), & \text{for each } S \in \rho_2^r, \\ x_e &\in \{0, 1\}, & \text{for each } e \in E^r, \end{aligned}$$

Note that f_2^r is intersecting supermodular on the set system ρ_2^r : Suppose $S_1, S_2 \in \rho_2^r$ such that $S_1 \cap S_2 \neq \emptyset$. Then $\overline{S_1} \cup \overline{S_2} = \overline{S_1 \cap S_2} \neq V$, and we have that $r \in \overline{S_1} \cap \overline{S_2} \neq \emptyset$, so the sets $\overline{S_1}$ and $\overline{S_2}$ are crossing, and from the crossing supermodularity of f we get

$$\begin{aligned} f_2(S_1) + f_2(S_2) &= f(\overline{S_1}) + f(\overline{S_2}) \leq f(\overline{S_1 \cap S_2}) + f(\overline{S_1 \cup S_2}) \\ &= f(\overline{S_1 \cap S_2}) + f(\overline{S_1 \cup S_2}) = f_2(S_1 \cup S_2) + f_2(S_1 \cap S_2) \end{aligned}$$

That is, $f_2(S)$ is intersecting supermodular on ρ_2^r , and hence the linear programming relaxation of (7) is TDI, and the integer program can be solved optimally. $\square$

We have the following simple *algorithm*: solve (6) and (7) optimally, and return the union of the optimal arc sets of the two problems as a solution to (3).

Theorem 1. *The algorithm given above returns a solution to (3) which is within a factor of 2 of the optimal.*

Proof. Let x^* be an optimal solution to (3); $\tilde{x}$ and $\hat{x}$ be optimal solutions to (6) and (7) respectively. Since x^* is a feasible solution for both (6) and (7), we have

$$2 * \sum_{e \in E} c_e x_e^* = \sum_{e \in E} c_e x_e^* + \sum_{e \in E} c_e x_e^* \geq \sum_{e \in E} c_e \tilde{x}_e + \sum_{e \in E} c_e \hat{x}_e. \quad (8)$$

Combining the optimal arc sets of (6) and (7) we will get a solution for (3) of cost at most $\sum_{e \in E} c_e \tilde{x}_e + \sum_{e \in E} c_e \hat{x}_e$, which is within a factor of 2 of the optimal. $\square$

3 The Description of the Basic Solutions

Consider the LP relaxation of the main problem (1).

$$\text{minimize } \sum_{e \in E} c_e x_e \quad (9)$$

subject to

$$\begin{aligned} x(\delta_G(S)) &\geq f(S), & \text{for each } S \subseteq V, \\ 0 &\leq x_e \leq 1, & \text{for each } e \in E. \end{aligned}$$

The second main result of the paper is the following property of the basic solutions of (9).

Theorem 2. *If $f(S)$ is crossing supermodular then in any basic solution of (9), $x_e \geq 1/4$ for at least one arc e .*

The rest of this section gives a proof to this theorem. The outline of the proof is analogous to Jain's [8] proof. Consider a basic solution x to the linear program.

- First we may assume without loss of generality that $x_e > 0$ for all arcs e . To see this simply delete all the arcs from the graph that have $x_e = 0$. Also assume $x_e < 1$ for all arcs e ; in the opposite case the theorem obviously holds.
- If x is a basic solution with m non-zero variables, then there must be m linearly independent inequalities in the problem that are satisfied by equation. Each inequality corresponds to a set S , and those satisfied by equations correspond to tight sets, i.e., sets S such that $x(\delta_G(S)) = f(S)$. We use the well-known structure of tight sets to show that there are m linearly independent such equations that correspond to a cross-free family of m tight sets. (A family of sets is cross-free if for all pairs of sets A and B in the family one of the four sets $A \setminus B$, $B \setminus A$, $A \cap B$ or the complement of $A \cup B$ is empty.)
- We will use the fact that a cross-free family of sets can be naturally represented by a forest.
- We will prove the theorem by contradiction. Assume that the statement is not true, and all variables have value $x_e < 1/4$. We consider any subgraph of the forest; using induction on the size of the subgraph we show that the k sets in this family have more than $2k$ endpoints. Applying this result for the whole forest we get there the m tight sets of the cross-free family have more than $2m$ endpoints. This is a contradiction since m arcs can have only $2m$ endpoints.

The hard part of the proof is the last step in this outline. While Jain can establish the same contradiction based on the assumption that all variables have $x_e < 1/2$, we will need the stronger assumption that $x_e < 1/4$ to reach a contradiction.

We start the proof by discussing the structure of tight sets. Call a set S *tight* if $x(\delta_G(S)) = f(S)$. Two sets A and B are called *intersecting* if $A \cap B$, $A \setminus B$, $B \setminus A$ are all nonempty. A family of sets is *laminar* if no two sets in it are intersecting, i.e., every pair of sets is either disjoint or one is contained in the other.

Two sets are called *crossing* if they are intersecting and $A \cup B$ is not the whole set V . A family of sets is *cross-free* if no two sets in the family are crossing. The key lemma for network design problems with crossing supermodular requirement function is that crossing tight sets have tight intersection and union, and the rows corresponding to these four sets in the constraint matrix are linearly dependent. Let $\mathcal{A}_G(S)$ denote the row corresponding to the set S in the constraint matrix of (1).

Lemma 2. *If two crossing sets A and B are tight then their intersection and union are also tight, and if $x_e > 0$ for all arcs then $\mathcal{A}_G(A) + \mathcal{A}_G(B) = \mathcal{A}_G(A \cup B) + \mathcal{A}_G(A \cap B)$.*

Proof. First observe that the function $x(\delta_G(S))$ is submodular, and that equation holds if and only if there are no positive weight arcs crossing from A to B or from B to A , i.e., if $x(\delta_G(A, B)) = x(\delta_G(B, A)) = 0$, where $\delta_G(A, B)$ denotes the arcs that are leaving A and entering B .

Next consider the chain of inequalities, using the crossing submodularity of $x(\delta_G(\cdot))$, supermodularity of f and the fact that A and B are tight.

$$\begin{aligned} f(A \cup B) + f(A \cap B) &\geq f(A) + f(B) = x(\delta_G(A)) + x(\delta_G(B)) \\ &\geq x(\delta_G(A \cup B)) + x(\delta_G(A \cap B)) \geq f(A \cup B) + f(A \cap B). \end{aligned}$$

This implies that both $A \cap B$ and $A \cup B$ are tight and $x(\delta_G(A, B)) = x(\delta_G(B, A)) = 0$. By our assumption that $x_e > 0$ on all arcs e , this implies that $\delta_G(A, B)$ and $\delta_G(B, A)$ are both empty, and so $\mathcal{A}_G(A) + \mathcal{A}_G(B) = \mathcal{A}_G(A \cup B) + \mathcal{A}_G(A \cap B)$. $\square$

The main tool for the proof is a cross-free family of $|E(G)|$ linearly independent tight sets.

Lemma 3. *For any basic solution x that has $x_e > 0$ for all arcs e , there exists a cross-free family Q of tight subsets of V satisfying the following:*

- $|Q| = |E(G)|$.
- The vectors $\mathcal{A}_G(S)$, $S \in Q$ are independent.
- For every set $S \in Q$, $f(S) \geq 1$.

Proof. For any basic solution x that has $x_e > 0$ for all arcs e there must be a set of $|E(G)|$ tight sets so that the vectors $\mathcal{A}_G(S)$ corresponding to the sets S are linearly independent.

Let Q be the family of all tight sets S . We use the technique of [7] to uncross sets in Q and to create the family of tight sets in the lemma. Note that the rank of the set of vectors $\mathcal{A}_G(S)$ is $|E(G)|$. Suppose there are two crossing tight sets A and B in the family. We can delete either of A or B from our family and add both the union and the intersection. This step maintains the properties that

- all the sets in our family are tight,
- the rank of the set of vectors $\mathcal{A}_G(S)$ for $S \in Q$ is $|E(G)|$.

This is true, as the union and intersection are tight by Lemma 2, and by the same lemma the vector of the deleted element B can be expressed as a linear combination of the other three as $\mathcal{A}_G(B) = \mathcal{A}_G(A \cup B) + \mathcal{A}_G(A \cap B) - \mathcal{A}_G(A)$.

In [7] it was shown that a polynomial sequence of such steps will result in a family whose sets are cross-free. Now we can delete dependent elements to get the family of sets satisfying the first two properties stated in the lemma.

To see the last property observe that the assumption that $x_e > 0$ for all arcs e implies that all the tight sets S that have a nonzero vector $\mathcal{A}_G(S)$ must have $f(S) \geq 1$. $\square$

Following the notation of András Frank we will represent a cross-free family as a laminar family with two kind of sets: round and square sets. Take any element v of $V(G)$. Let $S \in Q$ be a round set if $v \notin S$ and S be a square set if $v \in S$. It is easy to see that by representing all square sets by their complements we get a laminar family. This laminar family will provide the the natural forest representation for our cross-free family.

Lemma 4. *For any cross-free family Q , the sets $\{R \in Q \text{ for round sets}\}$, and the sets $\{V \setminus S \text{ for square sets } S \in Q\}$ together form a laminar family.*

The laminar family as given by Lemma 4 has the following *forest representation*. Let $R_1, \dots, R_k$ be the round sets and $S_1, \dots, S_l$ be the square sets of a cross-free family Q . Consider the corresponding laminar family $L = \{R_1, \dots, R_k, \bar{S}_1, \dots, \bar{S}_l\}$. Define the forest F as follows. The node set of F is L , and there is an arc from $U \in L$ to $W \in L$ if W is the smallest subset in L containing U . See the figure below for an example of a laminar family L obtained from a cross-free family Q and the tree representation of L .

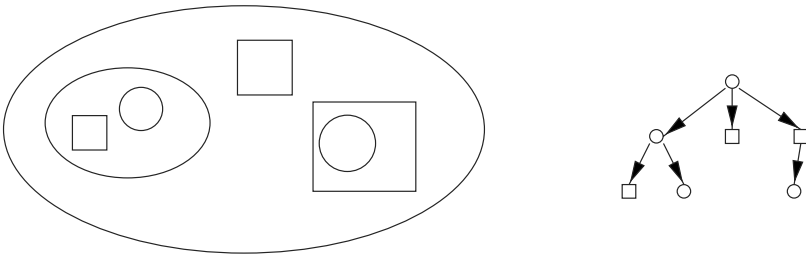


Fig. 1. A laminar family with tree representation

Now let's give some intuition for our main proof. Lemma 3 says that $|Q| = |E(G)|$, that is,

$$\text{There are exactly twice as many arc endpoints of } G \text{ as subsets of } Q \quad (10)$$

The idea of the proof comes from the following observation. Assuming that the statement of Theorem 1 is not true, that is for any $e \in E(G), x_e < 1/4$, distribute the endpoints such that each subset of Q gets at least 2 endpoints and some subsets get more than 2 endpoints. This will contradict (10). How to find this kind of distribution? The concept of incidence defined below gives the necessary connection between endpoints and subsets.

We say that an arc $e = i \rightarrow j$ of $G = (V, E)$ leaves a subset $U \in Q$ if $i \in U$ and $j \in \bar{U}$. Consider an arc $e = i \rightarrow j$ of $G = (V, E)$. We will define which set is the head and the tail of this arc incident to. If a node of the graph is the head or the tail of many arcs, then each head and tail at the same node may be incident to different sets. If $U \in L$ is the smallest round subset of L such that e is leaving

U then the tail of e at i is called an endpoint *incident* to U ; when no round subset satisfies the above condition then the tail of e at i is called incident to $W \in L$ if W is the smallest square subset containing both i and j . Similarly, if $U \in L$ is the smallest square subset of L such that e is leaving $\bar{U}$ then the head of e at j is called an endpoint *incident* to U ; when no square subset satisfies the above condition then the head of the arc e at j is called incident to $W \in L$ if W is the smallest round subset containing both i and j . For example, in the following picture, the tail at i is incident to S , head at j is incident to R , the head at w is incident to P , and the tail at u is not incident to any of given subsets.

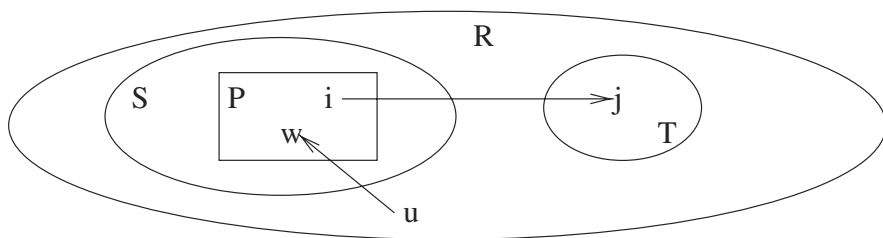


Fig. 2. Incidence between endpoints and subsets

Note that by definition each arc endpoint of G is incident to at most one subset of Q . Thus, in our distribution each subset can naturally get those endpoints which are incident to it. If a subset gets more than 2 endpoints, we will reallocate the “surplus” endpoints to some other subset “lacking” endpoints so that the minimum “quota” of 2 is provided for every subset.

How to do the reallocation? Here we can use the directed forest F defined above. That is, start allocating endpoints to the subsets which correspond to the leaves of the forest. If a subset has surplus endpoints reallocate them to its parent, call this process “pumping” the surplus endpoints up the tree. If at the end of the “pumping up” process, every non-root node gets 2 endpoints and at least one root node gets more than 2 endpoints then we are getting a contradiction to (10).

Obviously, the mathematical technique to accomplish this process of pumping up is induction. That is, achieve the above allocation first for smaller subtrees and using that achieve the allocation for larger subtrees. Ideally, we would like to prove that for any rooted subtree it is possible to allocate the endpoints such that each node gets 2 endpoints and the root gets at least some specified number k which is strictly greater than 2. Unfortunately, this kind of simple statement doesn’t allow us to apply the induction successfully. The point is that the roots of some tree structures can always get more than k endpoints; and if we use this loose estimate for those “surplus-rich” nodes then this might cause some of their ancestors to lack even the minimum quota of 2. That is why we need to define

some tree structures and for each of them state a different k as the minimum number of endpoints that the root of that particular subtree can get.

Next we define the necessary tree structures. See the figures for examples.

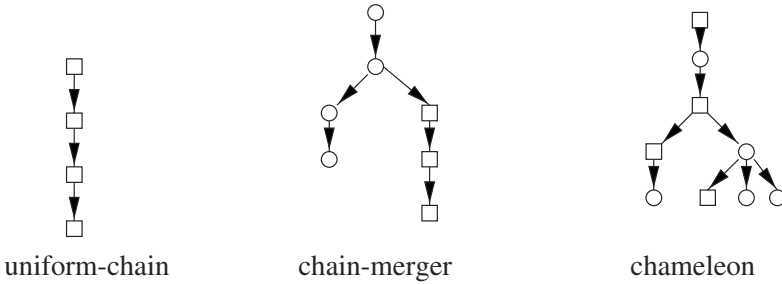


Fig. 3. Tree structures

A *chain* is a subtree of nodes, all of which are unary except maybe the *tail* (the lowest node in the chain). Note that a node of any arity is itself a chain.

A chain with a non-unary tail is called the *survival chain* of its *head* (the highest node in the chain). So if a node is not unary then its survival chain is the node itself.

A chain of nodes having the same shape and requirement 1 is called a *uniform-chain*.

A *chain-merger* is a union of a round-shape uniform-chain and a square-shape uniform-chain such that the head of one of the chains is a child of a node of the other chain.

A subtree is called *chameleon* if

- the root has one child;
- the survival chain of the root consists of nodes of both shapes;
- the tail of the survival chain has two children.

The rest of this section is the following main lemma which completes the proof of the theorem.

Lemma 5. *Suppose every arc takes value strictly less than $1/4$. Then for any rooted subtree of the forest, we can distribute the endpoints contained in it such that every vertex gets at least 2 endpoints and the root gets at least*

- 5 if the subtree is a uniform-chain;
- 6 if the subtree is a chameleon;
- 7 if the subtree is a chain-merger;
- 10 if the root has at least 3 children;
- 8 otherwise.

Proof. by induction on the height of the subtree.

First consider a *leaf* R . If the requirement of R is 1 then it is a uniform-chain and needs at least 5 endpoints allocated to it (hereafter, this allocated number will be called *label*). On the other hand, since all the variables have values less than $1/4$ then there are at least 5 arcs leaving R . Since R has no children this implies that there are at least 5 endpoints incident to it, so R gets label at least 5. If the requirement of R is greater than 1 then the same argument shows that it can get label at least 10.

For the subtrees having more than one node let's consider cases dependent on the number of children of the root R . Hereafter, without loss of generality we will assume that R is a *round node*; the other case is symmetric.

Case 1: If R has at least *four children*, by induction it gets label at least $4 \cdot 3 = 12$ since each child has excess of at least 3.

Case 2: Suppose R has *three children*: S_1, S_2, S_3 . If at least one of its children has label ≥ 6 then R will get at least $3 + 3 + 4 = 10$. Consider the case when all three children have labels 5, i.e., by induction the subtrees of all three are uniform-chains. Let's show that in this case R has an endpoint incident to it (and therefore gets label $\geq 3 + 3 + 3 + 1 = 10$). Consider cases dependent on the shapes of the children.

1. If the subtrees of S_1, S_2, S_3 are round-shaped uniform-chains then an arc leaving one of the S_i 's has its head in R . Otherwise, we would have $\mathcal{A}_G(R) = \mathcal{A}_G(S_1) + \mathcal{A}_G(S_2) + \mathcal{A}_G(S_3)$ which contradicts the independence of the vectors $\mathcal{A}_G(R), \mathcal{A}_G(S_1), \mathcal{A}_G(S_2), \mathcal{A}_G(S_3)$.
2. Suppose S_1 and S_2 are round-shaped and S_3 is square-shaped. Unlike the previous case, here some arcs which start from S_1 and S_2 might enter S_3 without creating endpoints for R .

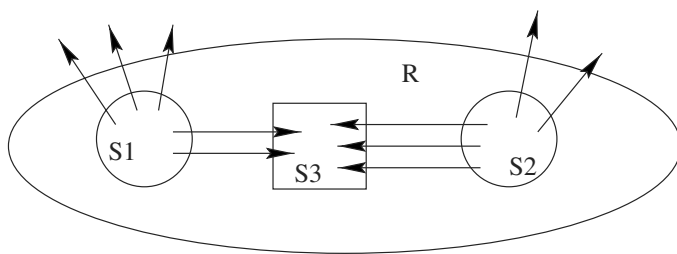


Fig. 4. Case 2.2

If R has no endpoints incident to it then $f(R) < f(S_1) + f(S_2) = 1 + 1 = 2$ and hence we must have $f(R) = 1$. Thus, the sum of the values of the arcs which leave S_1 and S_2 and enter S_3 , is equal to $f(S_1) + f(S_2) - f(R) = 1$. Hence, the requirement of 1 of $\bar{S}_3$ is completely satisfied by this kind of arcs.

In the result, $\mathcal{A}_G(R) = \mathcal{A}_G(S_1) + \mathcal{A}_G(S_2) - \mathcal{A}_G(\bar{S}_3)$ which contradicts the independence.

3. Suppose S_1 is round-shaped and S_2, S_3 are square-shaped. If there is no endpoint incident to R then all the arcs leaving S_1 should also leave R to satisfy the requirement of R which can be only 1; but this means $\mathcal{A}_G(R) = \mathcal{A}_G(S_1)$ contradicting the independence.
4. If S_1, S_2, S_3 are square-shaped then R should have endpoints to satisfy its positive requirement.

Case 3: Suppose R has *two children*, S_1 and S_2 . Then R needs label 7 if its subtree is a chain-merger and label 8 otherwise. Let's consider cases in that order.

1. The subtree of R is a chain-merger if and only if the subtrees of S_1 and S_2 are uniform-chains of different shapes. In this case, by the argument used in case 2.3, R has an endpoint incident to it, so it gets label at least $3+3+1=7$.
2. If none of the subtrees of S_1 and S_2 is a uniform-chain then R gets at least $4+4=8$. R gets at least $8=5+3$ also in the case when at least one of its children has label ≥ 7 .
3. The remaining hard case is when the subtree of S_1 is a uniform-chain with label 5 and the subtree of S_2 is a chameleon with label 6. Let's show that R has an endpoint incident to it and so gets label at least $3+4+1=8$. Assume that R doesn't have endpoints. Consider 2 cases:
 - (a) Suppose S_1 is round-shaped. Let T be the highest round-shaped node in the survival chain of S_2 (can be S_2 itself). If an arc leaving T doesn't leave R then its head is incident to R ; so all the arcs leaving T leave also R . Since $f(S_1) = 1, f(R) \in Z$ and there is no endpoint incident to R then either $f(R) = f(T) + 1$ or $f(R) = f(T)$. In the first case, all the arcs leaving S_1 should leave also R (see figure), and $\mathcal{A}_G(R) = \mathcal{A}_G(S_1) + \mathcal{A}_G(T)$; in the second case no arc leaves both S_1 and R , and $\mathcal{A}_G(R) = \mathcal{A}_G(T)$; so the independence is violated in both cases.

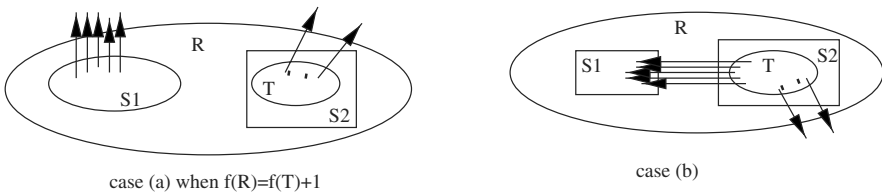
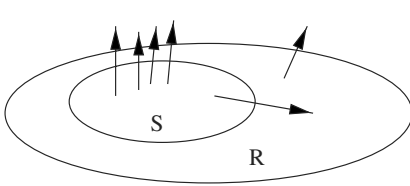


Fig. 5. Two children cases

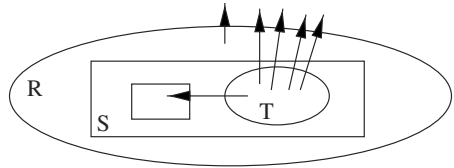
- (b) Suppose S_1 is square-shaped. Let T be the highest round-shaped node in the survival chain of S_2 . There is an arc leaving both T and $\bar{S}_1$, otherwise we would have $\mathcal{A}_G(R) = \mathcal{A}_G(T)$. In that case, all the arcs leaving $\bar{S}_1$ should leave also T because $f(\bar{S}_1) = 1$ and $f(R) \in Z$ (see figure). Thus, $\mathcal{A}_G(T) = \mathcal{A}_G(\bar{S}_1) + \mathcal{A}_G(R)$ contradicting the independence.

Case 4: Suppose R has *one child*, S . Consider cases dependent on the structure of the subtrees of R and S .

1. Suppose the subtree of R is a round-shaped *uniform-chain*, and hence R needs label 5. Then the subtree of S is also a uniform-chain. The independence of $\mathcal{A}_G(R)$ and $\mathcal{A}_G(S)$ (along with integral requirement of R) implies that R should have at least 2 endpoints incident to it (see figure); thus, R gets label $\geq 3 + 2 = 5$.



case 4.1



case 4.2(b)

Fig. 6. One child cases

2. Suppose the subtree of R is a *chameleon*, and hence R needs label 6. Consider 3 cases.
 - (a) The subtree of S is a chameleon. Then the survival chain of S contains a round node T . The independence of $\mathcal{A}_G(R)$ and $\mathcal{A}_G(T)$ implies that R should have at least 2 endpoints incident to it; thus, R gets label $\geq 4 + 2 = 6$.
 - (b) The subtree of S , τ is a chain-merger. Then τ has a round node T such that every other round node of τ is contained in T . The independence of $\mathcal{A}_G(R)$ and $\mathcal{A}_G(T)$ implies that R should have at least 1 endpoint incident to it (see figure); thus, R gets label $\geq 5 + 1 = 6$.
 - (c) In all other cases, S has label at least 8, so R gets at least 6.
3. Suppose the subtree of R is a *chain-merger*, and hence R needs label 7. Consider 2 cases.
 - (a) The subtree of S is a chain-merger itself. Then R and S are round-shaped, and the independence of $\mathcal{A}_G(R)$ and $\mathcal{A}_G(S)$ implies that R should have at least 2 endpoints incident to it. Thus, R gets label $\geq 5 + 2 = 7$.
 - (b) The subtree of S is a square-shaped uniform-chain. Since R contains only square-shaped subsets, it should have at least 5 endpoints to satisfy its requirement; thus, R gets $\geq 3 + 5 = 8$.

4. Suppose the subtree of R is *none of the 3 structures* considered above, and hence R needs label 8. Then the subtree of S can't be a uniform-chain or chameleon, and has label at least 7. Consider 2 cases.
 - (a) The survival chain of S contains a round-shaped subset T . Then the independence of $\mathcal{A}_G(R)$ and $\mathcal{A}_G(T)$ implies that R should have at least 2 endpoints incident to it. Thus, R gets label $\geq 7 + 2 = 9$.
 - (b) All the nodes in the survival chain of S are square-shaped. Then its tail T has at least 3 children and label ≥ 10 . Based on the same independence argument, any node in the survival-chain, including S , also gets label ≥ 10 . Thus, R gets at least 8.

□

4 The 4-Approximation Algorithm via Iterative Rounding

The idea of the algorithm is to iteratively round the solutions of the linear programs derived from the main IP formulation as described below.

Based on Theorem 2, if we include the arcs with value at least a quarter in the solution of the main integer program (1) then the factor that we lose in the cost is at most 4. These arcs might not form a feasible solution for (1) yet. We reduce the LP to reflect the set of edges included in the solution, and apply the method recursively.

Formally, let x^* be an optimal basic solution of (9) and $E_{1/4+}$ be the set of the arcs which take value at least $1/4$ in x^* . Let $E_{res} = E - E_{1/4+}$. Consider the residual graph $G_{res} = (V, E_{res})$ and the corresponding residual LP:

$$\text{minimize } \sum_{e \in E_{res}} c_e x_e \tag{11}$$

subject to

$$\begin{aligned} x(\delta_{G_{res}}(S)) &\geq f(S) - |E_{1/4+} \cap \delta_G(S)|, & \text{for each } S \subseteq V, \\ 0 &\leq x_e \leq 1, & \text{for each } e \in E_{res}. \end{aligned}$$

This residual program has the same structure as (9); the difference is that the graph G and the requirements $f(S)$ are reduced respectively to G_{res} and $f(S) - |E_{1/4+} \cap \delta_G(S)|$ considering that the arcs from $E_{1/4+}$ are already included in the integral solution. Theorem 2 can be applied to (11) if the reduced requirement function is crossing supermodular which is shown next.

A function $f : 2^V \mapsto Z$ is called submodular if $-f$ is supermodular, i.e., if for all $A, B \subseteq V$, $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$. The functions $|\delta_G(\cdot)|$ and more generally $x(\delta_G(\cdot))$ for any nonnegative vector x are the most classical examples of a submodular functions. The requirement function in the residual problem is the difference of a crossing supermodular function f and this submodular function, so it is also crossing supermodular.

Theorem 3. *Let $\tilde{G} = (V, \tilde{E})$ be a subgraph of the directed graph $G = (V, E)$. If $f : 2^V \mapsto Z$ is a crossing supermodular function then $f(S) - |\delta_{\tilde{G}}(S)|$ is also a crossing supermodular function.*

Summarizing, we have the following high-level description of our algorithm:

- Find an optimal basic solution to LP (9).
- Include all the arcs with values $1/4$ or more in the solution of (1).
- Delete all the arcs included in the solution from the graph, and solve the residual problem recursively.

The algorithm terminates when there are no positive requirements left.

The method requires solving the linear program (9). Note that (9) has a constraint for each subset S . Using the ellipsoid method Grötschel, Lovász and Schrijver [6] proved that such a linear program can be solved in polynomial time if there is a polynomial time separation subroutine, i.e., a method that for a given vector x can find a subset S such that $x(\delta_G(S)) < f(S)$ if such a set exists. Note that a violated set exists if and only if the minimum $\min_S (x(\delta_G(S)) - f(S))$ is negative. The function $x(\delta_G(S)) - f(S)$ is crossing submodular. Grötschel, Lovász and Schrijver [6] designed a polynomial time algorithm to minimize submodular functions. Note that a crossing submodular function is (fully) submodular if restricted to the sets $\{S : r \in S, v \notin S\}$ for any nodes $r \neq v$. We can obtain the minimum of a crossing submodular function by selecting a node r and computing the minimum separately over the sets $\{S : r \in S, v \notin S\}$ and $\{S : r \notin S, v \in S\}$ for all nodes $v \neq r$.

Theorem 4. *The iterative algorithm given above returns a solution for (1) which is within a factor of 4 of the optimal.*

Proof. We prove by induction that given a basic solution x^* to (9) the method finds a feasible solution to the integer program (1) of cost at most $4cx^*$. Consider an iteration of the method. We add the arcs $E_{1/4+}$ to the integer solution. The cost of these arcs is at most 4 times the corresponding part of the fractional solution x^* , i.e., $c(E_{1/4+}) \leq 4 \sum_{e \in E_{1/4+}} c_e x_e^*$. A feasible solution to the residual problem can be obtained by projecting the current solution x^* to the residual arcs. Using purification we obtain a basic solution to the residual linear program of cost at most the cost of x^* restricted to the arcs E_{res} . We recursively apply the method to find an integer solution to the residual problem of cost at most 4 times this cost, i.e., at most $4 \sum_{e \in E_{res}} c_e x_e^*$. This proves the theorem. $\square$

The algorithm stated above is assuming that we solve a linear program every iteration. However, as seen by the proof, it suffices to solve the linear program (9) once, and use purification to obtain a basic solution to the residual problems in subsequent iterations.

References

1. A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for generalized Steiner tree problems on networks. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 134–144, 1991.
2. M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. In *Proceedings of the 9th Annual Symposium of Discrete Algorithms*, pages 192–200, 1998.
3. J. Edmonds. Edge-disjoint branchings. In R. Rustin, editor, *Combinatorial Algorithms*, pages 91–96. Academic Press, New York, 1973.
4. M. X. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. P. Williamson. Approximation algorithms for network design problems. In *Proceedings of the 5th Annual Symposium on Discrete Algorithms*, pages 223–232, 1994.
5. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problem. *SIAM Journal on Computing*, 24:296–317, 1995.
6. M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, 1988.
7. C. A. Hurkens, L. Lovasz, A. Schrijver, and E. Tardos. How to tidy up your set-system? In *Proceedings of Colloquia Mathematica Societatis Janos Bolyai 52. Combinatorics*, pages 309–314, 1987.
8. K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. In *Proceedings of the 39th Annual Symposium on the Foundation of Computer Science*, pages 448–457, 1998.
9. S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *Journal of the Association of Computing Machinery*, 41:214–235, 1994.
10. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 475–484, 1997.
11. A. Frank. Kernel systems of directed graphs. *Acta Sci. Math (Szeged)*, 41:63–76, 1979.
12. D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15:435–454, 1995.

Optimizing over All Combinatorial Embeddings of a Planar Graph (Extended Abstract)

Petra Mutzel* and René Weiskircher**

Max-Planck-Institut für Informatik, Saarbrücken
mutzel@mpi-sb.mpg.de, weiski@mpi-sb.mpg.de

Abstract. We study the problem of optimizing over the set of all combinatorial embeddings of a given planar graph. Our objective function prefers certain cycles of G as face cycles in the embedding. The motivation for studying this problem arises in graph drawing, where the chosen embedding has an important influence on the aesthetics of the drawing. We characterize the set of all possible embeddings of a given biconnected planar graph G by means of a system of linear inequalities with $\{0, 1\}$ -variables corresponding to the set of those cycles in G which can appear in a combinatorial embedding. This system of linear inequalities can be constructed recursively using SPQR-trees and a new splitting operation. Our computational results on two benchmark sets of graphs are surprising: The number of variables and constraints seems to grow only linearly with the size of the graphs although the number of embeddings grows exponentially. For all tested graphs (up to 500 vertices) and linear objective functions, the resulting integer linear programs could be generated within 10 minutes and solved within two seconds on a Sun Enterprise 10000 using CPLEX.

1 Introduction

A graph is called planar when it admits a drawing into the plane without edge-crossings. There are infinitely many different drawings for every planar graph, but they can be divided into a finite number of equivalence classes. We call two planar drawings of the same graph *equivalent* when the sequence of the edges in clockwise order around each node is the same in both drawings. The equivalence classes of planar drawings are called *combinatorial embeddings*. A combinatorial embedding also defines the set of cycles in the graph that bound faces in a planar drawing.

The complexity of embedding planar graphs has been studied by various authors in the literature [4, 3, 5]. E.g., Bienstock and Monma have given polynomial time algorithms for computing an embedding of a planar graph that

* Partially supported by DFG-Grant Mu 1129/3-1, Forschungsschwerpunkt “Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen”

** Supported by the Graduiertenkolleg “Effizienz und Komplexität von Algorithmen und Rechenanlagen”

minimizes various distance functions to the outer face [4]. Moreover, they have shown that computing an embedding that minimizes the diameter of the dual graph is NP-hard.

In this paper we deal with the following optimization problem concerned with embeddings: Given a planar graph and a cost function on the cycles of the graph. Find an embedding Π such that the sum of the cost of the cycles that appear as face cycles in Π is minimized. When choosing the cost 1 for all cycles of length greater or equal to five and 0 for all other cycles, the problem is NP-hard [13].

Our motivation to study this optimization problem and in particular its integer linear programming formulation arises in graph drawing. Most algorithms for drawing planar graphs need not only the graph as input but also a combinatorial embedding. The aesthetic properties of the drawing often changes dramatically when a different embedding is chosen.

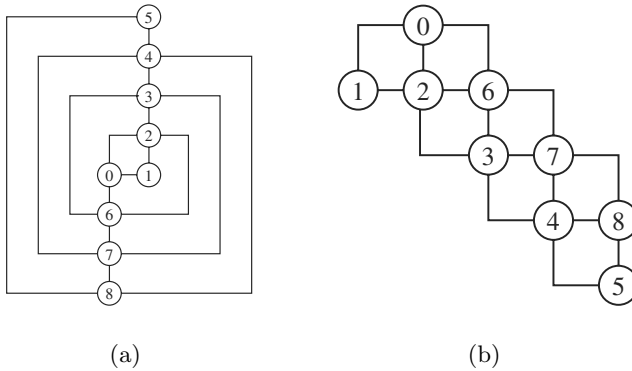


Fig. 1. The impact of the chosen planar embedding on the drawing

Figure 1 shows two different drawings of the same graph that were generated using the bend minimization algorithm by Tamassia [12]. The algorithm used different combinatorial embeddings as input. Drawing 1(a) has 13 bends while drawing 1(b) has only 7 bends. It makes sense to look for the embedding that will produce the best drawing. Our original motivation has been the following.

In graph drawing it is often desirable to optimize some cost function over all possible embeddings in a planar graph. In general these optimization problems are NP-hard [9]. For example: The number of bends in an orthogonal planar drawing highly depends on the chosen planar embedding. In the planarization method, the number of crossings highly depends on the chosen embedding when the deleted edges are reinserted into a planar drawing of the rest-graph. Both problems can be formulated as flow problems in the geometric dual graph. A flow between vertices in the geometric dual graph corresponds to a flow between

adjacent face cycles in the primal graph. Once we have characterized the set of all feasible embeddings (via an integer linear formulation on the variables associated with each cycle), we can use this in an ILP-formulation for the corresponding flow problem. Here, the variables consist of ‘flow variables’ and ‘embedding variables’.

This paper introduces an integer linear program whose set of feasible solutions corresponds to the set of all possible combinatorial embeddings of a given biconnected planar graph. One way of constructing such an integer linear program is by using the fact that every combinatorial embedding corresponds to a 2-fold complete set of cycles (see MacLane [11]). The variables in such a program are all simple cycles in the graph; the constraints guarantee that the chosen subset of all simple cycles is complete and that no edge of the graph appears in more than two simple cycles of the subset.

We have chosen another way of formulating the problem. The advantage of our formulation is that we only introduce variables for those simple cycles that form the boundary of a face in at least one combinatorial embedding of the graph, thus reducing the number of variables tremendously. Furthermore, the constraints are derived using the structure of the graph. We achieve this by constructing the program recursively using a data structure called SPQR-tree suggested by Di Battista and Tamassia ([1]) for the on-line maintenance of triconnected components. The static variant of this problem was studied in [10]. SPQR-trees can be used to code and enumerate all possible combinatorial embeddings of a biconnected planar graph. Furthermore we introduce a new splitting operation which enables us to construct the linear description recursively.

Our computational results on two benchmark sets of graphs have been quite surprising. We expected that the size of the linear system will grow exponentially with the size of the graph. Surprisingly, we could only observe a linear growth. However, the time for generating the system grows sub-exponentially; but for practical instances it is still reasonable. For a graph with 500 vertices and 10^{19} different combinatorial embeddings the construction of the ILP took about 10 minutes. Very surprising was the fact that the solution of the generated ILPs took only up to 2 seconds using CPLEX.

Section 2 gives a brief description of the data structure SPQR-tree. In Section 3 we describe the recursive construction of the linear constraint system using a new splitting operation. Our computational results are described in Section 4.

2 SPQR-Trees

In this section, we give a brief description of the SPQR-tree data structure for biconnected planar graphs. A connected graph is biconnected, if it has no cut vertex. A *cut vertex* of a graph $G = (V, E)$ is a vertex whose removal increases the number of connected components. A connected graph that has no cut vertex is called *biconnected*. A set of two vertices whose removal increases the number of connected components is called a *separation pair*; a connected graph without a separation pair is called *triconnected*.

SPQR-trees have been suggested by Di Battista and Tamassia ([1]). They represent a decomposition of a planar biconnected graph according to its split pairs. A *split pair* is a pair of nodes in the graph that is either connected by an edge or has the property that its removal increases the number of connected components. The *split components* of a split pair p are the maximal subgraphs of the original graph, for which p is not a split pair. When a split pair p is connected by an edge, one of the split components consists just of this edge together with the incident nodes while the other one is the original graph without the edge.

The construction of the SPQR-tree works recursively. At every node v of the tree, we split the graph into smaller edge-disjoint subgraphs. We add an edge to each of them to make sure that they are biconnected and continue by computing their SPQR-tree and making the resulting trees the subtrees of the node used for the splitting. Every node of the SPQR-tree has two associated graphs:

- The *skeleton* of the node defined by a split pair p is a simplified version of the whole graph where the split-components of p are replaced by single edges.
- The *pertinent graph* of a node v is the subgraph of the original graph that is represented by the subtree rooted at v .

The two nodes of the split pair p that define a node v are called the *poles* of v . For the recursive decomposition, a new edge between the poles is added to the pertinent graph of a node which results in a biconnected graph that may have multiple edges. The SPQR-tree has four different types of nodes that are defined by the structure and number of the split components of its poles v_a and v_b :

1. **Q-node:** The pertinent graph of the node is just the single edge $e = \{v_a, v_b\}$. The skeleton consists of the two poles that are connected by two edges. One of the edges represents the edge e and the other one the rest of the graph.
2. **S-node:** The pertinent graph of the node has at least one *cut vertex* (a node whose removal increases the number of connected components). When we have the cut vertices v_1, v_2 to v_k , they then split the pertinent graph into the components G_1, G_2 to G_{k+1} . In the skeleton of the node, G_1 to G_{k+1} are replaced by single edges and the edge between the poles is added. The decomposition continues with the subgraphs G_i , where the poles are v_i and v_{i+1} . Figure 2(a) shows the pertinent graph of an S-node together with the skeleton.
3. **P-node:** v_a and v_b in the pertinent graph have more than one split-components G_1 to G_k . In the skeleton, each G_i is replaced by a single edge and the edge between the poles is added. The decomposition continues with the subgraphs G_i , where the poles are again v_a and v_b . Figure 2(b) shows the pertinent graph of a P-node with 3 split components and its skeleton.
4. **R-node:** None of the other cases is applicable, so the pertinent graph is biconnected. The poles v_a and v_b are not a split pair of the pertinent graph. In this case, the decomposition depends on the *maximal split pairs* of the pertinent graph with respect to the pair $\{v_a, v_b\}$. A split pair $\{v_1, v_2\}$ is maximal with respect to $\{v_a, v_b\}$, if for every other split pair $\{v'_1, v'_2\}$, there

is a split component that includes the nodes v_1 , v_2 , v_a and v_b . For each maximal split pair p with respect to $\{v_a, v_b\}$, we define a subgraph G_p of the original graph as the union of all the split-components of p that do not include v_a and v_b . In the skeleton, each subgraph G_p is replaced by a single edge and the edge between the poles is added. The decomposition proceeds with the subgraphs defined by the maximal split pairs (see Fig. 2(c)).

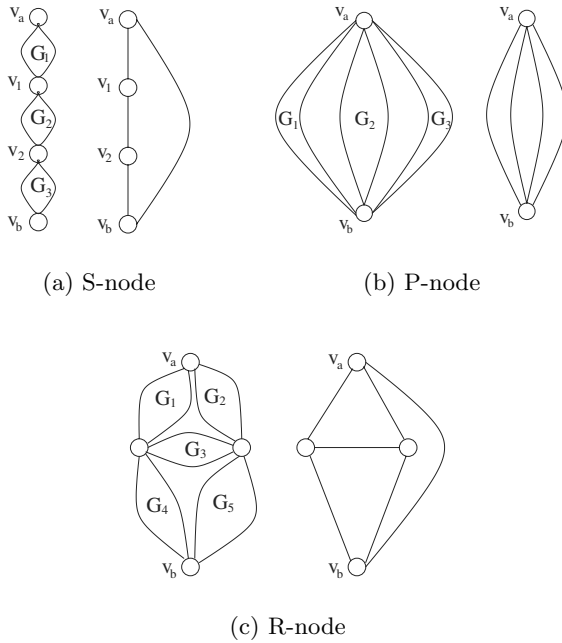


Fig. 2. Pertinent graphs and skeletons of the different node types of an SPQR-tree

The SPQR-tree of a biconnected planar graph G where one edge is marked (the so-called *reference edge*) is constructed in the following way:

1. Remove the *reference edge* and consider the end-nodes of it as the poles of the remaining graph G' . Depending on the structure of G' and the number of split components of the poles, choose the type of the new node v (S, P, R or Q).
2. Compute the subgraphs G_1 to G_k as defined above for the different cases and add an edge between the poles of each of the subgraphs.
3. Compute the SPQR-trees T_1 to T_k for the subgraphs where the added edge is the reference edge and make the root of these trees the sons of v .

When we have completed this recursive construction, we create a new Q-node representing the reference edge of G and make it the root of the whole SPQR-tree by making the old root a son of the Q-node. This construction implies that all leaves of the tree are Q-nodes and all inner nodes are S-, P-, or R-nodes. Figure 3 shows a biconnected planar graph and its SPQR-tree where the edge $\{1, 2\}$ was chosen as the reference edge.

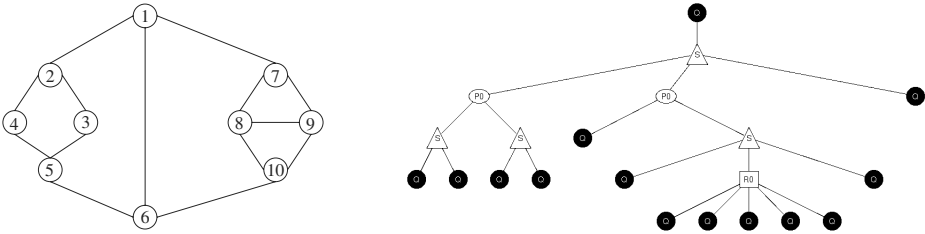


Fig. 3. A biconnected planar graph and its SPQR-tree

When we see the SPQR-tree as an unrooted tree, we get the same tree no matter what edge of the graph was marked as the reference edge. The skeletons of the nodes are also independent of the choice of the reference edge. Thus, we can define a unique SPQR-tree for each biconnected planar graph. Another important property of these trees is that their size (including the skeletons) is linear in the size of the original graph and they can be constructed in linear time ([1]).

As described in [1], SPQR-trees can be used to represent all combinatorial embeddings of a biconnected planar graph. This is done by choosing embeddings for the skeletons of the nodes in the tree. The skeletons of S- and Q-nodes are simple cycles, so they have only one embedding. The skeletons of R-nodes are always triconnected graphs. In most publications, combinatorial embeddings are defined in such a way, that only one combinatorial embedding for a triconnected planar graph exists (note that a combinatorial embedding does not fix the outer face of a drawing which realizes the embedding). Our definition distinguishes between two combinatorial embeddings which are mirror-images of each other (the order of the edges around each node in clockwise order is reversed in the second embedding). When the skeleton of a P-node has k edges, there are $(k-1)!$ different embeddings of its skeleton.

Every combinatorial embedding of the original graph defines a unique combinatorial embedding for each skeleton of a node in the SPQR-tree. Conversely, when we define an embedding for each skeleton of a node in the SPQR-tree, we define a unique embedding for the original graph. The reason for this fact is that each skeleton is a simplified version of the original graph where the split components of some split pair are replaced by single edges. Thus, if the SPQR-tree

of G has r R-nodes and the P-nodes P_1 to P_k where the skeleton of P_i has L_i edges, then the number of combinatorial embeddings of G is exactly

$$2^r \sum_{i=1}^k (L_i - 1)! .$$

Because the embeddings of the R- and P-nodes determine the embedding of the graph, we call these nodes the *decision nodes* of the SPQR-tree. In [2], the fact that SPQR-trees can be used to enumerate all combinatorial embeddings of a biconnected planar graph was used to devise a branch-and-bound algorithm for finding a planar embedding and an outer face for a graph such that the drawing computed by Tamassia's algorithm has the minimum number of bends among all possible orthogonal drawings of the graph.

3 Recursive Construction of the Integer Linear Program

3.1 The Variables of the Integer Linear Program

The skeletons of P-nodes are *multi-graphs*, so they have multiple edges between the same pair of nodes. Because we want to talk about directed cycles, we can be much more precise when we are dealing with *bidirected graphs*. A directed graph is called *bidirected* if there exists a bijective function $r : E \rightarrow E$ such that for every edge $e = (v, w)$ with $r(e) = e^R$ we have $e^R = (w, v)$ and $r(e^R) = e$. We can turn an undirected graph into a bidirected graph by replacing each undirected edge by two directed edges that go in opposite directions. The undirected graph G that can be transformed in this way to get the bidirected graph G' is called the *underlying graph* of G' .

A *directed cycle* in the bidirected graph $G = (V, E)$ is a sequence of edges of the following form: $c = ((v_1, v_2), (v_2, v_3), \dots, (v_k, v_1)) = (e_1, e_2, \dots, e_k)$ with the properties that every node of the graph is contained in at most two edges of c and if $k = 2$, then $e_1 \neq e_2$ holds. We say a planar drawing of a bidirected graph is the drawing of the underlying graph, so the embeddings of a bidirected graph are identical with the embeddings of the underlying graph.

A *face cycle* in a combinatorial embedding of a bidirected planar graph is a directed cycle of the graph, such that in any planar drawing that realizes the embedding, the left side of the cycle is empty. Note that the number of face cycles of a planar biconnected graph is $m - n + 2$ where m is the number of edges in the graph and n the number of nodes.

Now we are ready to construct an integer linear program (ILP) in which the feasible solutions correspond to the combinatorial embeddings of a biconnected planar bidirected graph. The variables of the program are binary and they correspond to directed cycles in the graph. As objective function, we can choose any linear function on the directed cycles of the graph. With every cycle c we associate a binary variable x_c . In a feasible solution of the integer linear program, a variable x_c has value 1 if the associated cycle is a face cycle in the represented

embedding and 0 otherwise. To keep the number of variables as small as possible, we only introduce variables for those cycles that are a face cycle in at least one combinatorial embedding of the graph.

3.2 Splitting an SPQR-Tree

We use a recursive approach to construct the variables and constraints of the ILP. Therefore, we need an operation that constructs a number of smaller problems out of our original problem such that we can use the variables and constraints computed for the smaller problems to compute the ILP for the original problem. This is done by splitting the SPQR-tree at some decision-node v . Let e be an edge incident to v whose other endpoint is not a Q-node. Deleting e splits the tree into two trees T_1 and T_2 . We add a new edge with a Q-node attached to both trees to replace the deleted edge and thus ensure that T_1 and T_2 become complete SPQR-trees again. The edges corresponding to the new Q-nodes are called *split edges*. For incident edges of v , whose other endpoint is a Q-node, the splitting is not necessary. Doing this for each edge incident to v results in $d + 1$ smaller SPQR-trees, called the *split-trees* of v , where d is the number of inner nodes adjacent to v . This splitting process is shown in Fig. 4. Since the new trees are SPQR-trees, they represent planar biconnected graphs which are called the *split graphs* of v . We will show how to compute the ILP for the original graph using the ILPs computed for the split graphs.

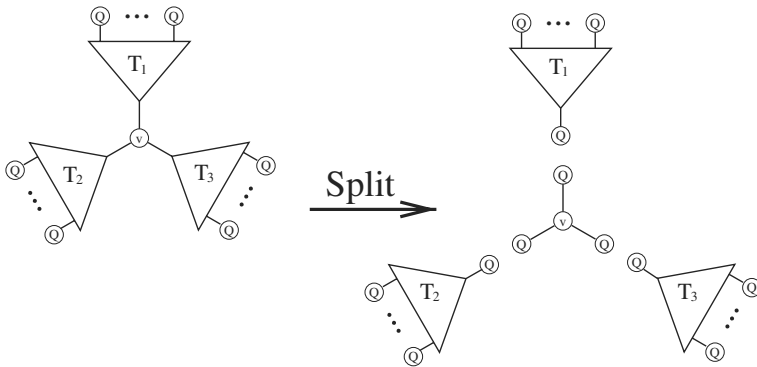


Fig. 4. Splitting an SPQR-tree at an inner node

As we have seen, the number and type of decision-nodes in the SPQR-tree of a graph determines the number of combinatorial embeddings. The subproblems we generated by splitting the tree either have only one decision-node or at least one fewer than the original problem.

3.3 The Integer Linear Program for SPQR-Trees with One Inner Node

We observe that a graph whose SPQR-tree has only one inner node is isomorphic to the skeleton of this inner node. So the split-tree of v which includes v , called the *center split-tree* of v , represents a graph which is isomorphic to the whole graph.

The ILPs for SPQR-trees with only one inner node are defined as follows:

- S-node: When the only inner node of the SPQR-tree is an S-node, the whole graph is a simple cycle. Thus it has two directed cycles and both are face-cycles in the only combinatorial embedding of the graph. So the ILP consists of two variables, both of which must be equal to one.
- R-node: In this case, the whole graph is triconnected. According to our definition of planar embedding, every triconnected graph has exactly two embeddings, which are mirror-images of each other. When the graph has m edges and n nodes, we have $k = 2(m - n + 2)$ variables and two feasible solutions. The constraints are given by the convex hull of the points in k -dimensional space, that correspond to the two solutions.
- P-node: The whole graph consists only of two nodes connected by k edges with $k \geq 3$. Every directed cycle in the graph is a face cycle in at least one embedding of the graph, so the number of variables is equal to the number of directed cycles in the graph. The number of cycles is $l = 2\binom{k}{2}$ because we always get an undirected cycle by pairing two edges and, since we are talking about directed cycles, we get twice the number of pairs of edges. As already mentioned, the number of embeddings is $(k - 1)!$. The constraints are given as the convex hull of the points in l -dimensional space that represent these embeddings.

3.4 Construction of the ILP for SPQR-Trees with More than One Inner Node

We define, how to construct the ILP of an SPQR-tree T from the ILPs of the split-trees of a decision node v of T . Let G be the graph that corresponds to T and $T_1, \dots, T_k$ the split-trees of v representing the graphs G_1 to G_k . We assume that T_1 is the center split-tree of v . Now we consider the directed cycles of G . We can distinguish two types:

1. *Local cycles* are cycles of G that also appear in one of the graphs $G_1, \dots, G_k$.
2. *Global cycles* of G are not contained in any of the G_i .

Every split-tree of v except the center split-tree is a subgraph of the original graph G with one additional edge (the split edge corresponding to the added Q-node). The graph that corresponds to the center split-tree may have more than one split edge. Note that the number of split edges in this graph is not necessarily equal to the degree of v , because v may have been connected to Q-nodes in the original tree. For every split edge e , we define a subgraph $expand(e)$

of the original graph G , which is represented by e . The two nodes connected by a split edge always form a split pair p of G . When e belongs to the graph G_i represented by the split-tree T_i , then $\text{expand}(e)$ is the union of all the split components of G that share only the nodes of p and no edge with G_i .

For every directed cycle c in a graph G_i represented by a split-tree, we define the set $R(c)$ of *represented cycles* in the original graph G . A cycle c' of G is in $R(c)$, when it can be constructed from c by replacing every split edge $e = (v, w)$ in c by a simple path in $\text{expand}(e)$ from v to w .

The variables of the ILPs of the split-trees that represent local cycles will also be variables of the ILP of the original graph G . But we will also have variables that correspond to global cycles of G . A global cycle c in G will get a variable in the ILP, when the following conditions are met:

1. There is a variable x_{c_1} in the ILP of T_1 with $c \in R(c_1)$.
2. For every split-tree T_i with $2 \leq i \leq k$ where c has at least one edge in G_i , there is a variable x_{c_i} in the ILP of T_i such that $c \in R(c_i)$.

So far we have defined all the variables for the integer linear program of G . The set C of all constraints of the ILP of T is given by

$$C = C_l \cup C_c \cup C_G \quad .$$

First we define the set C_l which is the set of *lifted constraints* of T . Each of the graphs $T_1, \dots, T_k$ is a simplified versions of the original graph G . They can be generated from G by replacing some split components of one or more split pairs by single edges. When we have a constraint that is valid for a split graph, a weaker version of this constraint is still valid for the original graph. The process of generating these new constraints is called *lifting* because we introduce new variables that cause the constraint to describe a higher dimensional half space or hyper plane. Let

$$\sum_{j=1}^l a_j x_{c_j} \doteq R$$

be a constraint in a split-tree, where $\doteq \in \{\leq, \geq, =\}$ and let X be the set of all variables of T . Then the lifted constraint for the tree T is the following:

$$\sum_{j=1}^l a_j \sum_{c: c \in R(c_j) \cap X} x_c \doteq R$$

We define C_l as the set of lifted constraints of all the split-trees. The number of constraints in C_l is the sum of all constraints in all split-trees.

The set C_c is the set of *choice constraints*. For a cycle c in G_i , which includes a split edge, we have $|R(c)| > 1$. All the cycles in $R(c)$ share either at least one directed edge or they pass a split graph of the split node in the same direction. Therefore, only one of the cycles in $R(c)$ can be a face cycle in any combinatorial

embedding of G (proof omitted). For each variable x_c in a split tree with $|R(c)| > 1$ we have therefore one constraint that has the following form:

$$\sum_{c' \in R(c) \wedge x_{c'} \in X} x_{c'} \leq 1$$

The set C_G consists of only one constraint, called the *center graph constraint*. Let F be the number of face cycles in a combinatorial embedding of G_1 , C_G the set of all global cycles c in G and C_L the set of all local cycles c in G_1 then this constraint is:

$$\sum_{c \in (C_g \cup C_l) \cap C} x_c = F$$

This constraint is valid, because we can produce every drawing D of G by replacing all split edges in a drawing D_1 of G_1 with the drawings of subgraphs of G . For each face cycle in D_1 , there will be a face cycle in D , that is either identical to the one in D_1 (if it was a local cycle) or is a global cycle. This defines the ILP for any biconnected planar graph.

3.5 Correctness of the ILP

Theorem 1. *Every feasible solution of the generated ILP corresponds to a combinatorial embedding of the given biconnected planar graph G and vice versa: every combinatorial embedding of G corresponds to a feasible solution for the generated ILP.*

Because the proof of the theorem is quite complex and the space is limited, we can only give a sketch of the proof. The proof is split into three lemmas.

Lemma 1. *Let G be a biconnected planar graph and let T be its SPQR-Tree. Let μ be a decision node in T with degree d , $T_1, \dots, T_{d'}$ with $d' \leq d$ be the split trees of μ (T_1 is the center split tree) and $G_1, \dots, G_{d'}$ the associated split graphs. Every combinatorial embedding Γ of G defines a unique embedding for each G_i . On the other side, if we fix a combinatorial embedding Γ_i for each G_i , we have defined a unique embedding for G .*

proof: (Sketch) To show the first part of the lemma, we start with a drawing Z of G that realizes embedding Γ . When G'_i is the graph G_i without its split edge, we get a drawing Z_1 of G_1 by replacing in Z the drawings of the G'_i with $2 \leq i \leq d'$ with drawings of single edges that are drawn inside the area of the plane formerly occupied by the drawing of G'_i . We can show that each drawing of G_1 that we construct in this way realizes the same embedding Γ_1 . We construct a planar drawing of each G_i with $2 \leq i \leq d'$ by deleting all nodes and edges from Z that are not contained in G_i and drawing the split edge into the area of the plane that was formerly occupied by the drawing of a path in G between the poles of G_i not inside G_i . Again we can show that all drawings produced in this way realize the same embedding Γ_i .

To show the second part of the lemma, we start with special planar drawings Z_i of the G_i that realize the embeddings Γ_i . We assume that Z_1 is a straight line drawing (such a drawing always exists [8]) and that each Z_i with $2 \leq i \leq d'$ is drawn inside an ellipse with the split edge on the outer face and the poles drawn as the vertices on the major axis of the ellipse. Then we can construct a drawing of G by replacing the drawings of the straight edges in Z_1 by the drawings Z_i of the G_i from which the split edges have been deleted. We can show that every drawing Z we construct in this way realizes the same embedding Γ of G . ■

To prove the main theorem, we first have to define the *incidence vector* of a combinatorial embedding. Let C be the set of all directed cycles in the graph that are face cycles in at least one combinatorial embedding of the graph. Then the incidence vector of an embedding Γ is given as a vector in $\{0, 1\}^{|C|}$ where the components representing the face cycles in Γ have value one and all other components have value zero.

Lemma 2. *Let $\Gamma = \{c_1, c_2, \dots, c_k\}$ be a combinatorial embedding of the biconnected planar graph G . Then the incidence vector χ^Γ satisfies all constraints of the ILP we defined.*

proof: (Sketch) We proof the lemma using induction over the number n of decision nodes in the SPQR-Tree T of G . The value $\chi(c)$ is the value of the component in χ associated with the cycle c . We don't consider the case $n = 0$, because G is a simple cycle in this case and has only one combinatorial embedding.

1. $n = 1$:

No splitting of the SPQR-tree is necessary, the ILP is defined directly by T . The variables are defined as the set of all directed cycles that are face cycles in at least one combinatorial embedding of G . Since the constraints of the ILP are defined as the convex hull of all incidence vectors of combinatorial embeddings of G , χ^Γ satisfies all constraints of the ILP.

2. $n > 1$:

From the previous lemma we know that Γ uniquely defines embeddings Γ_i with incidence vectors χ_i for the split graphs G_i . We will use the induction basis to show that χ^Γ satisfies all lifted constraints. We know that the choice constraints are satisfied by χ^Γ because in any embedding there can be only one cycle passing a certain split pair in the same direction. When lifting constraints, we replace certain variables by the sum of new variables and the choice constraints guarantee that this sum is either 0 or 1. Using this fact and the construction of the χ_i from χ^Γ , we can show that the sums of the values of the new variables are always equal to the value of the old variable. Therefore, all lifted constraints are satisfied.

To see that the center graph constraint is satisfied, we observe that any embedding of the skeleton of the split node has F faces. We can construct any embedding of G from an embedding Γ_1 of this skeleton by replacing edges by subgraphs. The faces in Γ that are global cycles are represented

by faces in Γ_1 and the faces that are local cycles in G are also faces in Γ_1 . Therefore the center graph constraint is also satisfied.

■

Lemma 3. *Let G be a biconnected planar graph and $\chi \in \{0, 1\}^{|C|}$ a vector satisfying all constraints of the ILP. Then χ is the incidence vector of a combinatorial embedding Γ of G .*

proof: Again, we use induction on the number n of decision nodes in the SPQR-tree T of G and we disregard the case $n = 0$.

1. $n = 1$:

Like in the previous lemma, our claim holds by definition of the ILP.

2. $n > 1$:

The proof works in two stages: First we construct vectors χ_i for each split graph from χ and prove that these vectors satisfy the ILPs for the G_i , and are therefore incidence vectors of embeddings Γ_i of the G_i by induction basis. In the second stage, we use the Γ_i to construct an embedding Γ for G and show that χ is the incidence vector of Γ .

The construction of the χ_i works as follows: When x is a variable in the ILP of G_i and the corresponding cycle is contained in G , then x gets the value of the corresponding variable in χ . Otherwise, we define the value of x as the sum of the values of all variables in χ whose cycles are represented by the cycle of x . This value is either 0 or 1 because χ satisfies the choice constraints.

Because χ satisfies the lifted constraints, the χ_i must satisfy the original constraints and by induction basis we know that each χ_i represents an embedding Γ_i of G_i . Using these embeddings for the split graphs, we can construct an embedding Γ for G like in lemma 1.

To show that χ is the incidence vector of Γ , we define χ^Γ as the incidence vector of Γ and show that χ and χ^Γ are identical. By construction of Γ and χ^Γ , the components in χ^Γ and χ corresponding to local cycles must be equal. The number of global cycles whose variable in χ has value 1 must be equal to the number of faces in Γ consisting of global cycles. This is guaranteed by the center graph constraint. Using the fact that for all face cycle in Γ_1 there must be a represented cycle in G whose component in χ and in χ^Γ is 1 we can show that both vectors agree on the values of the variables of the global cycles, and thus must be identical.

■

4 Computational Results

In our computational experiments, we tried to get statistical data about the size of the integer linear program and the times needed to compute it. Our implementation works for biconnected planar graphs with maximal degree four, since we

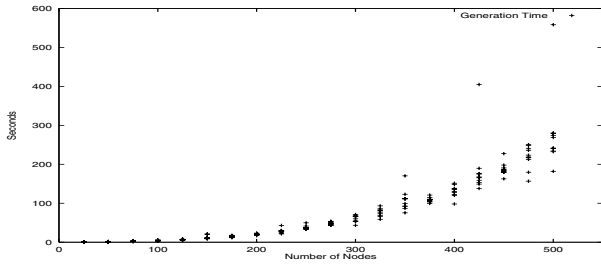


Fig. 5. Generation time for the ILP

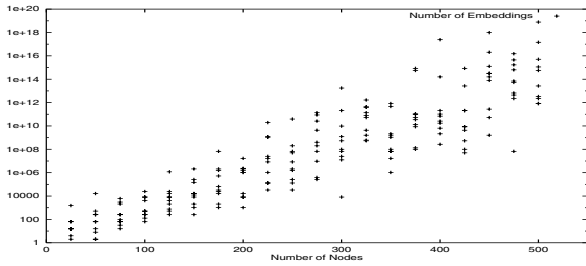


Fig. 6. Number of embeddings

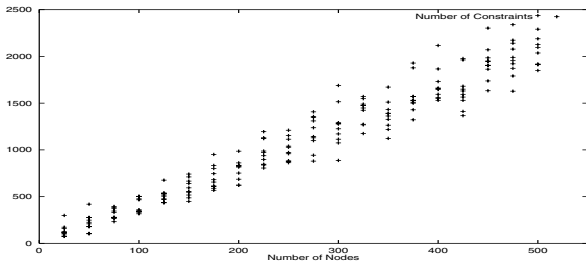


Fig. 7. Number of constraints

are interested in improving orthogonal planar drawings. First we used a benchmark set of 11491 practical graphs collected by the group around G. Di Battista in Rome ([6]). We have transformed these graphs into biconnected planar graphs with maximal degree four using planarization, planar augmentation, and the ring approach described in [7]. This is a commonly used approach getting orthogonal drawings with a small number of bends [7]. The obtained graphs have up to 160 vertices; only some of them had more than 100 different combinatorial embeddings. The maximum number of embeddings for any of the graphs was 5000.

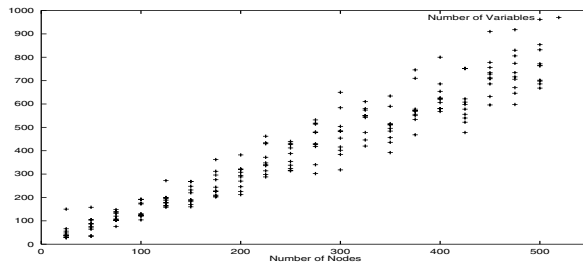


Fig. 8. Number of variables

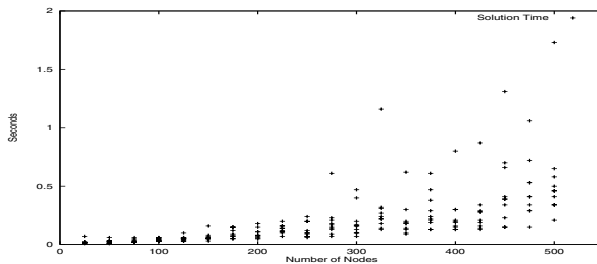


Fig. 9. Solution time

The times for generating the ILPs have been below one minute; the ILPs were quite small. CPLEX has been able to solve all of them very quickly.

In order to study the limits of our method, we started test runs on extremely difficult graphs. We used the random graph generator developed by the group around G. Di Battista in Rome that creates biconnected planar graphs with maximal degree four with an extremely high number of embeddings (see [2] for detailed information). We generated graphs with the number of nodes ranging from 25 to 500, proceeding in steps of 25 nodes and generating 10 random graphs for each number of nodes. For each of the 200 graphs, we generated the ILP and measured the time needed to do this. The times are shown in Fig. 5. They grow sub-exponentially and the maximum time needed was 10 minutes on a Sun Enterprise 10000.

The number of embeddings of each graph is shown in Fig. 6. They grow exponentially with the number of nodes, so we used a logarithmic scale for the y-axis. There was one graph with more than 10^{19} combinatorial embeddings. These numbers were computed by counting the number of R- and P-nodes in the SPQR-tree of each graph. Each R-node doubles the number of combinatorial embeddings while each P-node multiplies it by 2 or 6 depending on the number of edges in its skeleton. Figures 7 and 8 show the number of constraints and

variables in each ILP. Surprisingly, both of them grow linearly with the number of nodes. The largest ILP has about 2500 constraints and 1000 variables.

To test how difficult it is to optimize over the ILPs, we have chosen 10 random objective functions for each ILP with integer coefficients between 0 and 100 and computed a maximal integer solution using the mixed integer solver of CPLEX. Figure 9 shows the maximum time needed for any of the 10 objective functions. The computation time always stayed below 2 seconds.

Our future goal will be to extend our formulation such that each solution will not only represent a combinatorial embedding but an orthogonal drawing of the graph. This will give us a chance to find drawings with the minimum number of bends or drawings with fewer crossings. Of course, this will make the solution of the ILP much more difficult.

Acknowledgments We thank the group of G. Di Battista in Rome for giving us the opportunity to use their implementation of SPQR-trees in *GDToolkit*, a software library that is part of the ESPRIT ALCOM-IT project (work package 1.2), and to use their graph generator.

References

- [1] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25(5):956–997, October 1996.
- [2] P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *Lecture Notes in Computer Science*, 1272:331–344, 1998.
- [3] D. Bienstock and C. L. Monma. Optimal enclosing regions in planar graphs. *Networks*, 19(1):79–94, 1989.
- [4] D. Bienstock and C. L. Monma. On the complexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1):93–109, 1990.
- [5] J. Cai. Counting embeddings of planar graphs using DFS trees. *SIAM Journal on Discrete Mathematics*, 6(3):335–352, 1993.
- [6] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom. Theory Appl.*, 7:303–326, 1997.
- [7] P. Eades and P. Mutzel. *Algorithms and theory of computation handbook*, chapter 9 Graph drawing algorithms. CRC Press, 1999.
- [8] I. Fary. On straight line representing of planar graphs. *Acta. Sci. Math.(Szeged)*, 11:229–233, 1948.
- [9] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. *Lecture Notes in Computer Science*, 894:286–297, 1995.
- [10] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, August 1973.
- [11] S. MacLane. A combinatorial condition for planar graphs. *Fundamenta Mathematicae*, 28:22–32, 1937.
- [12] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.
- [13] G. J. Woeginger. personal communications, July 1998.

A Fast Algorithm for Computing Minimum 3-Way and 4-Way Cuts*

Hiroshi Nagamochi and Toshihide Ibaraki

Kyoto University, Kyoto, Japan 606-8501
{naga, ibaraki}@kuamp.kyoto-u.ac.jp

Abstract. For an edge-weighted graph G with n vertices and m edges, we present a new deterministic algorithm for computing a minimum k -way cut for $k = 3, 4$. The algorithm runs in $O(n^{k-2}(nF(n, m) + C_2(n, m) + n^2)) = O(mn^k \log(n^2/m))$ time for $k = 3, 4$, where $F(n, m)$ and $C_2(n, m)$ denote respectively the time bounds required to solve the maximum flow problem and the minimum 2-way cut problem in G . The bound for $k = 3$ matches the current best deterministic bound $\tilde{O}(mn^3)$ for weighted graphs, but improves the bound $\tilde{O}(mn^3)$ to $O(n(nF(n, m) + C_2(n, m) + n^2)) = O(\min\{mn^{8/3}, m^{3/2}n^2\})$ for unweighted graphs. The bound $\tilde{O}(mn^4)$ for $k = 4$ improves the previous best randomized bound $\tilde{O}(n^6)$ (for $m = o(n^2)$). The algorithm is then generalized to the problem of finding a minimum 3-way cut in a symmetric submodular system.

1 Introduction

Let $G = (V, E)$ stand for an undirected graph with a set V of vertices and a set E of edges being weighted by non-negative real numbers, and let n and m denote the numbers of vertices and edges, respectively. For an integer $k \geq 2$, a k -way cut is a partition $\{V_1, V_2, \dots, V_k\}$ of V consisting of k non-empty subsets. The problem of partitioning V into k non-empty subsets so as to minimize the weight sum of the edges between different subsets is called *the minimum k -way cut problem*. The problem has several important applications such as cutting planes in the traveling salesman problem [26], VLSI design [21], task allocation in distributed computing systems [20] and network reliability. The 2-way cut problem (i.e., the problem of computing the edge-connectivity) can be solved by $\tilde{O}(nm)$ time deterministic algorithms [7, 22] and by $\tilde{O}(n^2)$ and $\tilde{O}(m)$ time randomized algorithms [13, 14, 15]. For an unweighted planar graph, Hochbaum and Shmoys [9] proved that the minimum 3-way cut problem can be solved in $O(n^2)$ time. However, the complexity status of the problem for general $k \geq 3$ in an arbitrary graph G has been open for several years. Goldschmidt and Hochbaum proved that the problem is NP-hard if k is an input parameter [6]. In

* This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan, and the subsidy from the Inamori Foundation.

the same article [6], they presented an $O(n^{k^2/2-3k/2+4}F(n, m))$ time algorithm for solving the minimum k -way cut problem, where $F(n, m)$ denotes the time required to find a minimum (s, t) -cut (i.e., a minimum 2-way cut that separates two specified vertices s and t) in an edge-weighted graph with n vertices and m edges, which can be obtained by applying a maximum flow algorithm. This running time is polynomial for any fixed k . Afterwards, Karger and Stein [15] proposed a randomized algorithm that solves the minimum k -way cut problem with high probability in $O(n^{2(k-1)}(\log n)^3)$ time. For general k , a deterministic $O(n^{2k-3}m)$ time algorithm for the minimum k -way cut problem is claimed in [11] (where no full proof is available).

For $k = 3$, Kapoor [12] and Kamidoi *et al.* [10] showed that the problem can be solved in $O(n^3F(n, m))$ time, which was then improved to $\tilde{O}(mn^3)$ by Burlet and Goldschmidt [1]. For $k = 4$, Kamidoi *et al.* [10] gave an $O(n^4F(n, m)) = \tilde{O}(mn^5)$ time algorithm.

Let us call a non-empty and proper subset X of V a *cut*. Clearly, if we can identify the first cut V_1 in a minimum k -way cut $\{V_1, \dots, V_k\}$, then the rest of cuts $V_2, \dots, V_k$ can be obtained by solving the minimum $(k-1)$ -way cut problem in the graph induced by $V - V_1$. For $k = 3$, Burlet and Goldschmidt [1] succeeded to characterize a set of $O(n^2)$ number of cuts which contains at least one such cut V_1 . Thus, by solving $O(n^2)$ minimum $(k-1)$ -way cut problems, a minimum k -way cut can be computed. They showed that, given a minimum 2-way cut $\{X, V - X\}$ in G , such V_1 is a cut whose weight is less than $4/3$ of the weight of a minimum 2-way cut in G or in the induced subgraphs $G[X]$ and $G[V - X]$. Since it is known [24] that there are $O(n^2)$ cuts with weight less than $4/3$ of the weight of a minimum 2-way cut, and all those cuts can be enumerated in $\tilde{O}(mn^3)$ time, their approach yields an $\tilde{O}(mn^3)$ time minimum 3-way cut algorithm.

In this paper, we consider the minimum k -way cut problem for $k = 3, 4$, and give a new characterization of a set of $O(n)$ number of such candidate cuts for the first cut V_1 , on the basis of the submodularity of cut functions. We also show that those $O(n)$ cuts can be obtained in $O(n^2F(n, m))$ time by using Vazirani and Yannakakis's algorithm for enumerating small cuts [28]. Therefore, we can find a minimum 3-way cut in $O(n^2F(n, m) + nC_2(n, m)) = O(mn^3 \log(n^2/m))$ time and a minimum 4-way cut in $O(n^3F(n, m) + n^2C_2(n, m)) = O(mn^4 \log(n^2/m))$ time, where $C_2(n, m)$ denotes the time required to find a minimum 2-way cut in an edge-weighted graph with n vertices and m edges. The bound for $k = 3$ matches the current best deterministic bound $\tilde{O}(mn^3)$ for weighted graphs, but improves the bound $\tilde{O}(mn^3)$ to $O(\min\{mn^{8/3}, m^{3/2}n^2\})$ for unweighted graphs (since $F(n, m) = O(\min\{mn^{2/3}, m^{3/2}\})$ is known for unweighted graphs [2, 12]). The bound $\tilde{O}(mn^4)$ for $k = 4$ improves the previous best randomized bound $\tilde{O}(n^6)$ (for $m = o(n^2)$). In the case of an edge-weighted planar graph G , we also shows that the algorithm can be implemented to run in $O(n^3)$ time for $k = 3$ and in $O(n^4)$ time for $k = 4$, respectively. The algorithm is then generalized to the problem of finding a minimum 3-way cut in a symmetric submodular system.

In the next section, we review some basic results of cuts and symmetric submodular functions. In section 3, we present a new algorithm for computing

minimum 3-way and 4-way cuts in an edge-weighted graph. In section 4, we extend the algorithm to the problem of finding a minimum 3-way cut in a symmetric submodular system, and analyze its running time for the case in which the symmetric submodular function is a cut function of a hypergraph. Finally in section 5, we make some remarks on our approach in this paper.

2 Preliminaries

2.1 Notations and Definitions

A singleton set $\{x\}$ may be simply written as x , and “ $\subset$ ” implies proper inclusion while “ $\subseteq$ ” means “ $\subset$ ” or “ $=$ ”. For a finite set V , a *cut* is defined as a non-empty and proper subset X of V . For two disjoint subsets $S, T \subset V$, we say that a cut X *separates* S and T if $S \subseteq X \subseteq V - T$ or $T \subseteq X \subseteq V - S$ holds. A cut X *intersects* another cut Y if $X - Y \neq \emptyset$, $Y - X \neq \emptyset$ and $X \cap Y \neq \emptyset$ hold, and X *crosses* Y if, in addition, $V - (X \cup Y) \neq \emptyset$ holds. A set $\mathcal{X}$ of cuts of V is called *non-crossing* if no two cuts $X, Y \in \mathcal{X}$ cross each other (it is possible that $\mathcal{X}$ contains a pair of intersecting cuts).

The following observation plays a key role in analyzing the time bound of our algorithm.

Lemma 1. *Let V be a non-empty finite set, and $\mathcal{X}$ be a family of cuts in V such that, for each subset $X \in \mathcal{X}$, its complement $V - X$ does not belong to $\mathcal{X}$. Then $|\mathcal{X}| \leq 2|V| - 3$.*

Proof. Choose an arbitrary vertex $r \in V$ as a reference point, and assume that each cut $X \in \mathcal{X}$ does not contain r without loss of generality (replace X with $V - X$ if necessary). Now for any two cuts $X, Y \in \mathcal{X}$, $r \in V - (X \cup Y) \neq \emptyset$. Thus, $\mathcal{X}$ is a non-intersecting laminar of non-empty subsets of $V - r$. Then it is not difficult to see that $|\mathcal{X}| \leq 2|V - r| - 1 = 2|V| - 3$. $\square$

A *set function* f on a ground set V is a function $f : 2^V \mapsto \mathfrak{R}$, where $\mathfrak{R}$ is the set of real numbers. A set function f is called *submodular* if it satisfies the following inequality

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y) \text{ for every pair of subsets } X, Y \subseteq V. \quad (2.1)$$

An f is called *symmetric* if

$$f(X) = f(V - X) \text{ for all subsets } X \subseteq V. \quad (2.2)$$

For a symmetric and submodular set function f , it holds

$$f(X) + f(Y) \geq f(X - Y) + f(Y - X) \text{ for every pair of subsets } X, Y \subseteq V. \quad (2.3)$$

A pair (V, f) of a finite set V and a set function f on V is called a *system*. It is called a *symmetric submodular system* if f is symmetric and submodular.

Given a system (V, f) , we define the minimum k -way cut problem as follows. A k -way cut is a partition $\pi = \{V_1, V_2, \dots, V_k\}$ of V consisting of k non-empty subsets. The weight $\omega_f(\pi)$ of a k -way cut π is defined by

$$\omega_f(\pi) = \frac{1}{2}(f(V_1) + f(V_2) + \dots + f(V_k)). \quad (2.4)$$

A k -way cut is called *minimum* if it has the minimum weight among all k -way cuts in (V, f) .

Let $G = (V, E)$ be an undirected graph with a set V of vertices and a set E of edges weighted by non-negative reals. For a non-empty subset $X \subseteq V$, let $G[X]$ denote the graph induced from G by X . For a subset $X \subset V$, its *cut value*, denoted by $c(X)$, is defined to be the sum of weights of edges between X and $V - X$, where $c(\emptyset)$ and $c(V)$ are defined to be 0. This set function c on V is called the *cut function* of G . The cut function c is symmetric and submodular, as easily verified. For a k -way cut $\pi = \{V_1, V_2, \dots, V_k\}$ of V , its weight in G is defined by

$$\omega_c(\pi) = \frac{1}{2}(c(V_1) + c(V_2) + \dots + c(V_k)),$$

which means the weight sum of the edges between different cuts in π .

2.2 Enumerating All 2-Way Cuts

In [28], Vazirani and Yannakakis presented an algorithm that finds all the 2-way cuts in G in the order of non-decreasing weights. The algorithm finds the next 2-way cut by solving at most $2n - 3$ maximum flow problems. We describe this result in a slightly more general way in terms of set functions. This algorithm will be used in Section 3 to obtain minimum 3-way and 4-way cuts.

Theorem 2. [28] *For a system (V, f) with $n = |V|$, 2-way cuts in (V, f) can be enumerated in the order of non-decreasing weights with $O(nF_f)$ time delay between two consecutive outputs, where F_f is the time required to find a minimum 2-way cut that separates specified two disjoint subsets $S, T \subset V$ in (V, f) .*

Proof. Suppose that, for specified two disjoint subsets $S, T \subset V$, there is a procedure $A(S, T)$ for finding a minimum 2-way cut that separates S and T in F_f time. Let $V = \{v_1, \dots, v_n\}$. To represent a set of 2-way cuts, we use a p -dimensional $\{0, 1\}$ -vector μ , where p satisfies $1 \leq p \leq n$ and $\mu(i)$ denotes the i -th entry of vector μ . We assume that $\mu(1) = 1$ for all the vectors μ in the subsequent discussion. A p -dimensional $\{0, 1\}$ -vector μ gives $S_\mu = \{v_i \mid i \in \{1, \dots, p\}, \mu(i) = 1\}$ and $T_\mu = \{v_i \mid i \in \{1, \dots, p\}, \mu(i) = 0\}$ (i.e., $p = |S_\mu \cup T_\mu|$ holds), and represents the set $\mathcal{C}(\mu)$ of all 2-way cuts $\{X, V - X\}$ such that

$$S_\mu \subseteq X \subseteq V - T_\mu$$

(i.e., those separating S_μ and T_μ). In particular, $\mathcal{C}(\mu_1)$ with a 1-dimensional vector μ_1 represents all 2-way cuts $\{X, V - X\}$ (with $v_1 \in X$), while $\mathcal{C}(\mu_n)$ with

an n -dimensional vector μ_n consists of a single 2-way cut $\{S_\mu, T_\mu\}$, where S_μ and T_μ are specified by the vector $\mu = \mu_n$.

We first note that, if we remove the single cut in $\mathcal{C}(\mu_n)$ represented by an n -dimensional vector μ_n from the entire set $\mathcal{C}(\mu_1)$ of 2-way cuts, the set of remaining 2-ways cuts can be represented by the union of $(n - 1)$ sets $\mathcal{C}(\overline{\mu_n^p})$, $p = 2, 3, \dots, n$, such that $\overline{\mu_n^p}$ is the p -dimensional vector defined by

$$\overline{\mu_n^p}(i) = \begin{cases} \mu_n(i) & \text{if } 1 \leq i \leq p - 1 \\ 1 - \mu_n(i) & \text{if } i = p. \end{cases}$$

(Note that there is at most one vector $\overline{\mu_n^p}$ in which all entries are 1.)

Given a p -dimensional vector μ , where $2 \leq p \leq n$, we can find a minimum 2-way cut $\{Y, V - Y\}$ over all the 2-way cuts in $\mathcal{C}(\mu)$ in F_f time by applying procedure $A(S_\mu, T_\mu)$ if $T_\mu \neq \emptyset$, i.e., μ contains at least one 0-value entry (recall that $v_1 \in S_\mu$); otherwise (if $T_\mu = \emptyset$, i.e., $\mu(i) = 1$ for all $i \in \{1, \dots, p\}$), such a minimum 2-way cut $\{Y, V - Y\}$ can be found by applying procedure $A(S_\mu, T)$ at most $(n - 2)$ times by choosing $T = \{v_i\}$ for all $v_i \in V - S_\mu$. Let $\mu^*(\mu)$ denote the n -dimensional $\{0, 1\}$ -vector that represents the minimum 2-way cut $\{Y, V - Y\} \in \mathcal{C}(\mu)$ obtained by this procedure, and let $f(\mu)$ be its weight.

With these notations, an algorithm for enumerating all 2-way cuts in the order of non-decreasing weights can now be described. Initially we compute $\mu^*(\mu_1)$ and $f(\mu_1)$ for the 1-dimensional vector μ_1 with $\mu_1(1) = 1$. Let $Q := \{\mu_1\}$. Then we can enumerate 2-way cuts one after another in the non-decreasing order by repeatedly executing the next procedure B .

Procedure B

Choose a vector $\alpha \in Q$ with the minimum weight $f(\alpha)$; $\mu_n := \mu^*(\alpha)$;

Output μ_n ; $Q := Q - \{\alpha\}$;

Let a be the dimension of α , and for each $\beta = \overline{\mu_n^p}$, $p = a + 1, a + 2, \dots, n$, compute $\mu^*(\beta)$ and $f(\beta)$, and $Q := Q \cup \{\beta\}$.

Notice that after deleting α from Q , we do not have to add vectors $\overline{\mu_n^1}, \dots, \overline{\mu_n^a}$ to Q at this point (since these vectors have already been generated and added to Q). It is not difficult to see that the procedure B correctly finds the next smallest 2-way cut. The procedure B computes at most $n - 1$ $\mu^*(\beta)$, each of which can be obtained by procedure A once, except for the case where all entries of β are 1 (in this case $\mu^*(\beta)$ is computed by applying procedure A $n - 1$ times). By maintaining set $\text{argmin}\{f(\alpha) \mid \alpha \in Q\}$ by using a data structure of heap, a vector $\alpha \in Q$ with the minimum $f(\alpha)$ can be obtained in $O(\log |Q|) = O(\log(2^n)) = O(n)$. Thus the time delay to find the next smallest 2-way cut is $O((n - 2 + n - 1)(F_f + n)) = O(nF_f)$ (assuming $F_f = \Omega(n)$). $\square$

It is known that a non-empty and proper subset X that minimizes $g(X)$ in a submodular system (V, g) can be found in polynomial time by using the ellipsoid method [5]. In particular, the problem of finding a minimum 2-way cut which separates specified two disjoint subsets $S, T \subset V$ in (V, f) can be solved in polynomial time, since the problem is reduced to finding a minimum 2-way

cut in the system (V', g) such that $V' = V - (S \cup T)$ and a submodular function $g : V' \rightarrow \mathbb{R}$ defined by $g(X) = f(X \cup S) - f(S)$.

For the cut function c in an edge-weighted graph G , it is well known that a minimum 2-way cut that separates specified two disjoint subsets $S, T \subset V$ can be found by solving a single maximum flow problem. Therefore $F_f = O(mn \log(n^2/m))$ holds if we use the maximum flow algorithm of [4], where n and m are the numbers of vertices and edges in G , respectively. In the planar graph case, we can enumerate cuts more efficiently by making use of the shortest path algorithm in dual graphs.

Call a simple path between s and t an s, t -path, and let $S(n, m)$ denote the time to compute a shortest s, t -path in an edge-weighted graph G with n vertices and m edges.

Lemma 3. *For an edge-weighted graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, cycles in G can be enumerated in the order of non-decreasing lengths, with $O(S(n, m))$ time delay between two consecutive outputs, where the first cycle can be found in $O(mS(n, m))$ time.*

Proof. It is known [17] that s, t -paths can be enumerated in the order of non-decreasing lengths with $O(S(n, m))$ time delay between two consecutive outputs. Based on this, we enumerate cycles in G in the order of non-decreasing lengths as follows. Let $E = \{e_1 = (s_1, t_1), e_2 = (s_2, t_2), \dots, e_m = (s_m, t_m)\}$. Then the set $\mathcal{S}$ of all cycles in G can be partitioned into m sets $\mathcal{S}_i$, $i = 1, \dots, m$, of cycles, where $\mathcal{S}_i$ is the set of cycles $C \subseteq E$ such that $e_i \in C \subseteq E - \{e_1, \dots, e_{i-1}\}$. Also, a cycle $C \in \mathcal{S}_i$ is obtained by combining an s_i, t_i -path $P \subseteq E - \{e_1, \dots, e_{i-1}\}$ and the edge $e_i = (s_i, t_i)$ and vice versa. The shortest cycle in G can be obtained as a cycle with the minimum length among $P_i^* \cup \{e_i\}$, $i = 1, \dots, m$, where P_i^* is the shortest s_i, t_i -path in the graph $(V, E - \{e_1, \dots, e_{i-1}\})$. If $P_j^* \cup \{e_j\}$ is chosen as a cycle of the minimum length, then we compute the next shortest s_j, t_j -path P in the graph $(V, E - \{e_1, \dots, e_{j-1}\})$, and update P_j^* by this P . Then the second shortest cycle in G can be as a cycle with the minimum length among $P_i^* \cup \{e_i\}$, $i = 1, \dots, m$. Thus, by updating P_j^* by the next shortest s_j, t_j -path P after $P_j^* \cup \{e_j\}$ is chosen, we can repeatedly enumerate cycles in the order of non-decreasing weights, with $O(S(n, m))$ time delay. $\square$

Corollary 4. *For an edge-weighted planar graph $G = (V, E)$ with $n = |V|$, 2-way cuts in G can be enumerated in the order of non-decreasing weights with $O(n)$ time delay between two consecutive outputs, where the first 2-way cut can be found in $O(n^2)$ time.*

Proof. It is known that a 2-way cut in a planar graph $G = (V, E)$ corresponds to a cycle in its dual graph $G^* = (V^*, E)$, where G and G^* have the same edge set E . Thus, it suffices to enumerate cycles in G^* in the order of non-decreasing lengths, where the length of a cycle $C \subseteq E$ in G^* is the sum of weights of edges in C . By Lemma 3 (see Appendix), cycles in G^* can be enumerated in the order of non-decreasing lengths with $O(S(|V^*|, |E|))$ time delay. For a planar graph G^* , $S(|V^*|, |E|) = O(|V^*| + |E|) = O(|V|)$ is known [8]. $\square$

3 Minimum 3-Way and 4-Way Cuts

Let $\pi = \{V_1, \dots, V_k\}$ be a minimum k -way cut in a system (V, f) . Suppose that the first cut V_1 in π is known. Then we need to compute a minimum k -way cut $\{V_1, \dots, V_k\}$ in (V, f) under the restriction that V_1 is fixed. In the case of a graph, the problem can be reduced to finding a minimum $(k-1)$ -way cut in the induced subgraph $G[V - V_1]$. For $k = 3, 4$, we now show that such a cut V_1 can be obtained from among $O(n)$ number of cuts.

Theorem 5. *For a symmetric submodular system (V, f) with $n = |V| \geq 4$, let*

$$\{X_1, V - X_1\}, \{X_2, V - X_2\}, \dots, \{X_r, V - X_r\}$$

be the first r smallest 2-way cuts in the order of non-decreasing weights, where r is the first integer r such that X_r crosses some X_q ($1 \leq q < r$). Let us denote $Y_1 = X_q - X_r$, $Y_2 = X_q \cap X_r$, $Y_3 = X_r - X_q$, and $Y_4 = V - (X_q \cup X_r)$. Then:

- (i) *There is a minimum 3-way cut $\{V_1, V_2, V_3\}$ of (V, f) such that $V_1 = X_i$ or $V_1 = V - X_i$ for some $i \in \{1, 2, \dots, r-1\}$, or $\{V_1, V_2, V_3\} = \{Y_j, Y_{j+1}, V - (Y_j \cup Y_{j+1})\}$ for some $j \in \{1, 2, 3, 4\}$ (where $Y_{j+1} = Y_1$ for $j = 4$).*
- (ii) *There is a minimum 4-way cut $\{V_1, \dots, V_4\}$ of (V, f) such that $V_1 = X_i$ or $V_1 = V - X_i$ for some $i \in \{1, 2, \dots, r-1\}$, or $\{V_1, \dots, V_4\} = \{Y_1, Y_2, Y_3, Y_4\}$.*

Proof. We have $f(X_1) \leq f(X_2) \leq \dots \leq f(X_r)$ by assumption.

(i) Let a minimum 3-way cut in G be denoted by $\{V_1, V_2, V_3\}$ with $f(V_1) \leq \min\{f(V_2), f(V_3)\}$. If $f(V_1) < f(X_r)$, this implies that $V_1 = X_i$ or $V_1 = V - X_i$ for some $i < r$. Therefore assume $f(V_1) \geq f(X_r)$. Thus, $\omega_f(\{V_1, V_2, V_3\}) \geq \frac{3}{2}f(X_r)$ holds. Now the cuts X_q and X_r cross each other, and $f(X_q) \leq f(X_r)$ holds by $q < r$. By (2.3), we have

$$f(X_q) + f(X_r) \geq f(X_q - X_r) + f(X_r - X_q).$$

Hence at least one of $f(X_q - X_r)$ and $f(X_r - X_q)$ is equal to or less than $\max\{f(X_q), f(X_r)\}$ ($= f(X_r)$). Thus,

$$f(Y_i) \leq f(X_r) \text{ holds for } i = 1 \text{ or } i = 3.$$

Assume that $f(X_h - X_j) \leq f(X_r)$ holds for $\{h, j\} = \{q, r\}$. Similarly, we obtain $\min\{f(X_q \cap X_r), f(X_q \cup X_r)\} \leq f(X_r)$ by (2.1). Thus,

$$f(Y_i) \leq f(X_r) \text{ holds for } i = 2 \text{ or } i = 4$$

(note that $f(X_q \cup X_r) = f(V - (X_q \cup X_r))$). Therefore, there is an index $j \in \{1, 2, 3, 4\}$ such that $f(Y_j) \leq f(X_r)$ and $f(Y_{j+1}) \leq f(X_r)$ (where Y_{j+1} implies Y_1 for $j = 4$). Clearly, $f(V - (Y_j \cup Y_{j+1}))$ is equal to $f(X_q)$ or $f(X_r)$. Therefore, for the 3-way cut $\{Y_j, Y_{j+1}, V - (Y_j \cup Y_{j+1})\}$, we have

$$\omega_f(\{Y_j, Y_{j+1}, V - (Y_j \cup Y_{j+1})\}) \leq \frac{3}{2}f(X_r) \leq \omega_f(\{V_1, V_2, V_3\}).$$

This implies that $\{Y_j, Y_{j+1}, V - (Y_j \cup Y_{j+1})\}$ is also a minimum 3-way cut.

(ii) Let a minimum 4-way cut in G be denoted by $\{V_1, V_2, V_3, V_4\}$ with $f(V_1) \leq \min\{f(V_2), f(V_3), f(V_4)\}$. Assume that $f(V_1) \geq f(X_r)$ holds, since otherwise (i.e., $f(V_1) < f(X_r)$), we are done. Thus, $2f(X_r) \leq 2f(V_1) \leq \omega_f(\{V_1, V_2, V_3, V_4\})$. From inequalities (2.1) - (2.3), we then obtain

$$2f(X_r) \geq f(X_q) + f(X_r) \geq \frac{1}{2}[f(X_q - X_r) + f(X_r - X_q) + f(X_q \cap X_r) + f(V - (X_q \cup X_r))].$$

Therefore, we have $\frac{1}{2}[f(X_q - X_r) + f(X_r - X_q) + f(X_q \cap X_r) + f(V - (X_q \cup X_r))] \leq f(X_q) + f(X_r) \leq 2f(X_r) \leq \omega_f(\{V_1, V_2, V_3, V_4\})$, indicating that $\{X_q - X_r, X_r - X_q, X_q \cap X_r, V - (X_q \cup X_r)\}$ is also a minimum 4-way cut in (V, f) . $\square$

Now we are ready to describe a new algorithm for computing minimum 3-way and 4-way cuts in an edge-weighted graph G . In the algorithm, minimum 2-way cuts will be stored in $\mathcal{X}$ in the non-decreasing order, until $\mathcal{X}$ becomes crossing, and an $O(n)$ number of k -way cuts will be stored in $\mathcal{C}$, from which a k -way cut of the minimum weight is chosen.

MULTIWAY

Input: An edge-weighted graph $G = (V, E)$ with $|V| \geq 4$ and an integer $k \in \{3, 4\}$.

Output: A minimum k -way cut π .

```

1   $\mathcal{X} := \mathcal{C} := \emptyset$ ;  $i := 1$ ;
2  while  $\mathcal{X}$  is non-crossing do
3    Find the  $i$ -th minimum 2-way cut  $\{X_i, V - X_i\}$  in  $G$ ;
4     $\mathcal{X} := \mathcal{X} \cup \{X_i\}$ ;
5    if  $|V - X_i| \geq k - 1$  then find a minimum  $(k - 1)$ -way cut  $\{Z_1, \dots, Z_{k-1}\}$ 
      in the induced subgraph  $G[V - X_i]$ , and add the  $k$ -way cut
       $\{X_i, Z_1, \dots, Z_{k-1}\}$  to  $\mathcal{C}$ ;
6    if  $|X_i| \geq k - 1$  then find a minimum  $(k - 1)$ -way cut  $\{Z'_1, \dots, Z'_{k-1}\}$ 
      in the induced subgraph  $G[X_i]$ , and add the  $k$ -way cut
       $\{Z'_1, \dots, Z'_{k-1}, V - X_i\}$  to  $\mathcal{C}$ ;
7     $i := i + 1$ 
8  end; /* while */
9  Let  $X_r$  be the last cut added to  $\mathcal{X}$ , and choose a cut  $X_q \in \mathcal{X}$  that crosses
       $X_r$ , where we denote  $Y_1 = X_q - X_r$ ,  $Y_2 = X_q \cap X_r$ ,  $Y_3 = X_r - X_q$ , and
       $Y_4 = V - (X_q \cup X_r)$ ;
10 if  $k = 3$  then add to  $\mathcal{C}$  3-way cuts  $\{Y_j, Y_{j+1}, V - (Y_j \cup Y_{j+1})\}$ ,
       $j \in \{1, 2, 3, 4\}$  (where  $Y_{j+1} = Y_1$  for  $j = 4$ );
11 if  $k = 4$  then add 4-way cut  $\{Y_1, Y_2, Y_3, Y_4\}$  to  $\mathcal{C}$ ;
12 Output a  $k$ -way cut  $\pi$  in  $\mathcal{C}$  with the minimum weight.
```

The correctness of algorithm MULTIWAY follows from Theorem 5. To analyze the running time of MULTIWAY, let $C_k(n, m)$ ($k \geq 2$) denote the time required to find a minimum k -way cut in an edge-weighted graph with n vertices and m edges.

By Lemma 1, the number r of iterations of the while-loop is at most $2n - 2 = O(n)$. Therefore, lines 5 and 6 in MULTIWAY requires $O(nC_{k-1}(n, m))$

time in total. The total time required to check whether $\mathcal{X}$ is crossing or not is $O(rn^2) = O(n^3)$. By Theorem 2, r minimum 2-way cuts can be enumerated in the non-decreasing order in $O(rnF(n, m)) = O(n^2F(n, m))$ time, where $F(n, m)$ denotes the time required to find a minimum (s, t) -cut (i.e., a minimum 2-way cut that separates two specified vertices s and t) in an edge-weighted graph with n vertices and m edges. Summing up these, we have

$$C_3(n, m) = O(n^2F(n, m) + nC_2(n, m) + n^3),$$

$$C_4(n, m) = O(n^2F(n, m) + nC_3(n, m) + n^3) = O(n^3F(n, m) + n^2C_2(n, m) + n^4),$$

establishing the following result.

Theorem 6. *For an edge-weighted graph $G = (V, E)$, where $n = |V| \geq 4$ and $m = |E|$, a minimum 3-way cut and a minimum 4-way cut can be computed in $O(n^2F(n, m) + nC_2(n, m) + n^3)$ and $O(n^3F(n, m) + n^2C_2(n, m) + n^4)$ time, respectively. $\square$*

Currently, $F(n, m) = O(mn \log(n^2/m))$ [4] and $O(\min\{n^{2/3}, m^{1/2}\} m \log(n^2/m) \log U)$ (where U is the maximum edge weight if weights are all integers) [3] are among the fastest, and $C_2(n, m) = O(mn \log(n^2/m))$ [7] are known for weighted graphs. Thus our bounds become

$$C_3(n, m) = O(mn^3 \log(n^2/m)) \text{ or } O(\min\{n^{2/3}, m^{1/2}\} mn \log(n^2/m) \log U),$$

$$C_4(n, m) = O(mn^4 \log(n^2/m)) \text{ or } O(\min\{n^{2/3}, m^{1/2}\} mn^2 \log(n^2/m) \log U).$$

For unweighted graphs, we have $C_3(n, m) = O(\min\{mn^{2/3}, m^{3/2}\}n)$ and $C_4(n, m) = O(\min\{mn^{2/3}, m^{3/2}\}n^2)$, since $F(n, m) = O(\min\{mn^{2/3}, m^{3/2}\})$ is known for unweighted graphs [2,12]. For a planar graph G , we can obtain better bounds by applying Corollary 4.

Corollary 7. *For an edge-weighted planar graph $G = (V, E)$, where $n = |V| \geq 4$, a minimum 3-way cut and a minimum 4-way cut can be computed in $O(n^3)$ and $O(n^4)$ time, respectively. $\square$*

4 3-Way Cuts in Symmetric Submodular Systems

4.1 General Case

Let (V, f) be a symmetric submodular system. Let us consider whether the algorithm MULTIWAY can be extended to find a minimum k -way cut in (V, f) for $k = 3, 4$. In the case of a graph $G = (V, E)$, if the first cut V_1 in a minimum k -way cut $\{V_1, \dots, V_k\}$ is known, then the remaining task of finding cuts $V_2, \dots, V_k$ can be reduced to computing a minimum $(k-1)$ -way cut in the induced subgraph $G[V - V_1]$. However, even if the first cut V_1 in a minimum k -way cut $\{V_1, \dots, V_k\}$ is known in (V, f) , finding the rest of cuts $V_2, \dots, V_k$ may not be directly reduced to the minimum $(k-1)$ -way cut problem in a certain symmetric submodular system $(V - V_1, f')$ (because, we do not have any concept corresponding to the

induced subgraph $G[V - V_1]$ in a symmetric submodular system (V, f) . Thus, we need to find a procedure to compute a minimum k -way cut $\{V_1, \dots, V_k\}$ in (V, f) under the restriction that V_1 is fixed. In what follows, we show that such a procedure can be constructed for $k = 3$ (but the case of $k = 4$ is still open).

For this, we first define a set function f' on $V' = V - V_1$ from the cut V_1 .

Lemma 8. *For a symmetric submodular system (V, f) and a subset $V_1 \subset V$, the set function f' on $V' = V - V_1$, defined by*

$$f'(X) = f(X) + f(V' - X) - f(V'), \quad X \subseteq V',$$

is symmetric and submodular.

Proof. Clearly f' is symmetric by definition. For any $X, Y \subseteq V'$, we obtain

$$\begin{aligned} f'(X) + f'(Y) &= f(X) + f(V' - X) - f'(V) + f(Y) + f(V' - Y) - f'(V) \\ &\geq f(X \cap Y) + f(X \cup Y) + f(V' - (X \cap Y)) + f(V' - (X \cup Y)) - 2f'(V) \\ &\quad (\text{by the submodularity of } f) \\ &= f'(X \cap Y) + f'(X \cup Y). \square \end{aligned}$$

Note that for a 3-way cut $\pi = \{V_1, V_2, V_3\}$ in (V, f) , its weight $\omega_f(\pi)$ is given by $\frac{1}{2}(f(V_1) + f(V_2) + f(V - V_1 - V_2)) = f(V_1) + \frac{1}{2}(f(V_2) + f(V' - V_2) - f(V_1)) = f(V_1) + \frac{1}{2}f'(V_2)$ for $V' = V - V_1$. Therefore, a minimum 3-way cut $\{V_1, V_2, V_3\}$ in (V, f) for a specified V_1 can be obtained by solving the minimum 2-way cut problem in the system (V', f') .

For a symmetric submodular system (V, f) , it is known that a minimum 2-way cut in a symmetric submodular system can be computed in $O(n^3 T_f)$ time [27, 23], where T_f is the time to evaluate the function value $f(X)$ for a given subset $X \subseteq V$.

Now we are ready to extend the algorithm MULTIWAY so as to find a minimum 3-way cut in (V, f) . By Theorem 2, we can enumerate 2-way cuts of (V, f) in the order of non-decreasing weights with $O(nF_f)$ time delay, where F_f is the time required to find a minimum 2-way cut that separates specified two disjoint subsets $S, T \subset V$ in (V, f) . Thus, we can enumerate the first r minimum 2-way cuts of (V, f) in $O(rnF_f) = O(n^2 F_f)$ time, using the result $r = O(n)$ of Lemma 1. Therefore, we can compute a minimum 3-way cut $\{V_1, V_2, V_3\}$ in (V, f) under the restriction that V_1 is fixed. Summarizing these up, we have the following result.

Theorem 9. *For a symmetric and submodular system (V, f) with $n = |V| \geq 3$, a minimum 3-way cut can be computed in $O(n^2 F_f + n^4 T_f)$ time, where F_f is the time required to find a minimum 2-way cut that separates specified two disjoint subsets $S, T \subset V$ in (V, f) , and T_f is the time to evaluate the function value $f(X)$ for a given subset $X \subseteq V$. $\square$*

4.2 Cut Function of a Hypergraph

Now we elaborate upon the result of Theorem 9, assuming that f is the cut function c of a hypergraph. Let $H = (V, E)$ be an edge-weighted hypergraph

with a vertex set V and a hyper-edge set E , where a hyper-edge e is defined as a non-empty subset of V . For a hyper-edge $e \in E$, $V(e)$ denotes the set of end vertices of e (i.e., the vertices incident to e). A *cut function* c in H is defined by

$$c(X) = \{w(e) \mid V(e) \cap X \neq \emptyset, V(e) \cap (V - X) \neq \emptyset\}, \quad X \subseteq V,$$

where $w(e)$ is the weight of edge e . It is easy to observe that the cut function c in a hypergraph H is also symmetric and submodular. We define the weight $\omega_c(\pi)$ of a k -way cut $\{V_1, \dots, V_k\}$ also by (2.4).

Remark: The weight $\omega_c(\pi)$ of a minimum 2-way cut π is equal to the minimum weight sum of edges whose removal makes the hypergraph disconnected. This is the same as the case of the cut function in a graph. For $k \geq 3$, however, $\omega_c(\pi)$ of a minimum k -way cut $\pi = \{V_1, \dots, V_k\}$ may not mean the minimum weight sum γ_H of edges whose removal generates at least k components, where

$$\gamma_H = \min\{\gamma_H(\pi) \mid k\text{-way cut } \pi\}, \quad (4.5)$$

$$\gamma_H(\pi) = \{w(e) \mid e \in E, V(e) \not\subseteq V_i \text{ for all } V_i \in \pi\}.$$

This is different from the case of a graph. The reason is because the weight $w(e)$ of a hyper-edge e may contribute to $\omega_c(\pi)$ by more than 1 if e is incident to more than two subsets in π . $\square$

Let $H = (V, E)$ be a hypergraph with $|E| \geq |V|$. Given two specified vertices $s, t \in V$, it is known [19] that a minimum 2-way cut separating s and t in H can be computed by solving a single maximum flow problem in the auxiliary digraph with $|V| + 2|E|$ vertices and $2d_H + 2|E|$ edges, where

$$d_H = \sum_{e \in E} |V(e)|.$$

Thus, a minimum 2-way cut separating two disjoint subsets $S, T \subset V$ can be computed in $F_c = F(|V| + 2|E|, 2d_H + 2|E|) = \tilde{O}(|E|d_H)$ time by computing a minimum 2-way cut separating s and t in the hypergraph obtained from H by contracting S and T into single vertices s and t , respectively.

For a given subset $X \subset V' = V - V_1$, the function value $f'(X) = c(X) + c(V' - X) - c(V_1)$ can be evaluated in $T_c = O(n + d_H)$ time. Hence, by Theorem 9, we can compute a minimum 3-way cut in $O(|V|^2 F_c + |V|^4 T_c) = \tilde{O}(|V|^2 |E| d_H + |V|^4 (|V| + d_H)) = \tilde{O}(|V|^4 + |V|^2 |E| d_H)$ time.

The running time can be slightly improved as follows. Notice that for a 3-way cut $\pi = \{V_1, V_2, V_3\}$ in $H = (V, E)$, its weight $\omega_c(\pi)$ is written as

$$\begin{aligned} \omega_c(\pi) &= \sum \{w(e) \mid e \in E, |\{i \mid V_i \cap V(e) \neq \emptyset\}| = 2\} \\ &\quad + \frac{3}{2} \sum \{w(e) \mid e \in E, |\{i \mid V_i \cap V(e) \neq \emptyset\}| = 3\}. \end{aligned}$$

For a given cut V_1 , let us define an edge-weighted hypergraph $H[V_1] = (V', E')$ induced from H by V_1 by

$$V' = V - V_1 \text{ and } E' = \{e \in E \mid V(e) \cap (V - V_1) \neq \emptyset\},$$

where for each hyper-edge $e \in E'$, the set $V'(e)$ of its end vertices and its weight $w'(e)$ are redefined by $V'(e) = V(e) - V_1$ and

$$w'(e) = \begin{cases} w(e) & \text{if } V'(e) = V(e) \text{ (i.e., } V(e) \cap V_1 = \emptyset) \\ w(e)/2 & \text{if } V'(e) \subset V(e) \text{ (i.e., } V(e) \cap V_1 \neq \emptyset). \end{cases}$$

Hence, for a 2-way cut $\pi' = \{V_2, V_3\}$ in $H[V_1] = (V', E')$, its weight $\omega_{c'}(\pi') = \sum\{w'(e) \mid e \in E', V'(e) \cap V_2 \neq \emptyset \neq V'(e) \cap V_3\}$ (where c' denotes the cut function in $H[V_1]$) is equal to $\sum\{w(e) \mid e \in E', V(e) \cap V_2 \neq \emptyset \neq V(e) \cap V_3, V(e) \cap V_1 = \emptyset\} + \sum\{w(e)/2 \mid e \in E', V'(e) \cap V_2 \neq \emptyset, V'(e) \cap V_3 \neq \emptyset, V'(e) \cap V_1 \neq \emptyset\}$. Thus, $c(V_1) + \omega_{c'}(\pi') = \omega_c(\pi)$ holds for a 3-way cut $\pi = \{V_1, V_2, V_3\}$ and the cut function c in H . Therefore, for a fixed cut V_1 , we can find a minimum 3-way cut $\{V_1, V_2, V_3\}$ in H by computing a minimum 2-way cut $\{V_2, V_3\}$ in $H[V_1]$. It is known that a minimum 2-way cut in a hypergraph $H = (V, E)$ can be obtained in $O(|V|d_H + |V|^2 \log |V|)$ time [18]. Since we need to solve $O(n)$ number of minimum 2-way cut problems in our algorithm, the time complexity becomes $O(|V|^2 F(|V| + |E|, d_H) + |V|^2 (d_H + |V| \log |V|)) = \tilde{O}(|V|^2 |E| d_H)$, assuming $|E| \geq |V|$.

Corollary 10. *For an edge-weighted hypergraph $H = (V, E)$, a 3-way cut π minimizing $\omega_c(\pi)$ can be computed in $O(n^2 m d_H)$ time, where $n = |V|$, $m = |E| \geq n$, $d_H = \sum_{e \in E} |V(e)|$ and c is the cut function in H . $\square$*

As remarked in the above, the weight of $\omega_c(\pi)$ of a minimum 3-way cut π in a hypergraph $H = (V, E)$ may not give the γ_H of (4.5). However, for any 3-way cut π , it holds

$$\omega_c(\pi) \leq \frac{3}{2} \gamma_H(\pi).$$

Therefore, the minimum value $\omega_c(\pi)$ over all 3-way cuts π is at most 1.5 of γ_H .

Corollary 11. *For an edge-weighted hypergraph $H = (V, E)$, there is a 1.5-approximation algorithm for the problem of minimizing the sum of weights of hyper-edges whose removal leaves at least 3 components. The algorithm runs in $\tilde{O}(n^2 m d_H)$ time, where $n = |V|$, $m = |E|$ and $d_H = \sum_{e \in E} |V(e)| \geq n$. $\square$*

For $k = 4$, it is not known how to find a minimum 4-way cut $\{V_1, \dots, V_4\}$ in a symmetric submodular system (or in a hypergraph) under the condition that V_1 is fixed. By using the above argument, we can find a minimum 4-way cut in a hypergraph $H = (V, E)$ if every hyper-edge e has at most *three* incident vertices (i.e., $2 \leq |V(e)| \leq 3$ for all $e \in E$). This is because if $|V(e)| \leq 3$ for all $e \in E$, then it holds $\omega_c(\pi) = c(V_1) + \omega_{c'}(\pi')$ for the weight $\omega_c(\pi)$ of 4-way cut $\pi = \{V_1, V_2, V_3, V_4\}$ in H and the weight $\omega_{c'}(\pi')$ of a 3-way cut $\pi' = \{V_2, V_3, V_4\}$ in the induced hypergraph $H[V_1]$ as defined above. The algorithm runs in $\tilde{O}(|V|^2 F_c +$

$|V|(|V|^2|E|d_H)) = \tilde{O}(|V|^3|E|d_H)$ time. Also, in this class of hypergraphs, a 4-way cut π that minimizes the weight $\omega_c(\pi)$ is a 1.5-approximation to the problem of finding a 4-way cut π that minimizes $\gamma_H(\pi)$.

Corollary 12. *Let $H = (V, E)$ be an edge-weighted hypergraph such that every hyper-edge $e \in E$ has at most three incident vertices, and let $n = |V|$, $m = |E| \geq n$ and $d_H = \sum_{e \in E} |V(e)|$. Then a 4-way cut π that minimizes $\omega_c(\pi)$ can be computed in $\tilde{O}(n^3 m d_H)$ time, and its $\gamma_H(\pi)$ value is at most 1.5 of the minimum weight sum γ_H of hyper-edges whose removal leaves at least 4 components in H .* $\square$

5 Concluding Remark

In this paper, we proposed an algorithm for computing minimum 3-way and 4-way cuts in a graph, based on the enumeration algorithm of 2-way cuts in the non-decreasing order. The algorithm is then extended to the problem of finding a minimum 3-way cut in a symmetric submodular system. As a result of this, the minimum 3-way cut problem in a hypergraph can be solved in polynomial time. It is left for future work how to extend the algorithm to the case of $k \geq 5$ in graphs and the case of $k \geq 4$ in symmetric submodular systems (or in hypergraphs). Very recently, we proved that the minimum k -way cuts in graphs can be computed in $O(mn^k \log(n^2/m))$ time for $k = 5, 6$ [25].

Acknowledgments

This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan, and the subsidy from the Inamori Foundation. We would like to thank Professor Naoki Katoh and Professor Satoru Iwata for their valuable comments.

References

1. M. Burlet and O. Goldschmidt, *A new and improved algorithm for the 3-cut problem*, Operations Research Letters, vol.21, (1997), pp. 225–227.
2. S. Even and R. E. Tarjan, *Network flow and testing graph connectivity*, SIAM J. Computing, vol.4, (1975), pp. 507–518.
3. A. V. Goldberg and S. Rao, *Beyond the flow decomposition barrier*, Proc. 38th IEEE Annual Symp. on Foundations of Computer Science, (1997), pp. 2–11.
4. A. V. Goldberg and R. E. Tarjan, *A new approach to the maximum flow problem*, J. ACM, vol.35, (1988), pp. 921–940.
5. M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, Berlin (1988).
6. O. Goldschmidt and D. S. Hochbaum, *A polynomial algorithm for the k -cut problem for fixed k* , Mathematics of Operations Research, vol.19, (1994), pp. 24–37.
7. J. Hao and J. Orlin, *A faster algorithm for finding the minimum cut in a directed graph*, J. Algorithms, vol. 17, (1994), pp. 424–446.

8. M. R. Henzinger, P. Klein, S. Rao and D. Williamson, *Faster shortest-path algorithms for planar graphs*, J. Comp. Syst. Sc., vol. 53, (1997), pp. 2-23.
9. D. S. Hochbaum and D. B. Shmoys, *An $O(|V|^2)$ algorithm for the planar 3-cut problem*, SIAM J. Algebraic Discrete Methods, vol.6, (1985), pp. 707-712.
10. Y. Kamidoi, S. Wakabayashi and N. Yoshida, *Faster algorithms for finding a minimum k -way cut in a weighted graph*, Proc. IEEE International Symposium on Circuits and Systems, (1997), pp. 1009-1012.
11. Y. Kamidoi, S. Wakabayashi and N. Yoshida, *A new approach to the minimum k -way partition problem for weighted graphs*, Technical Report of Inst. Electron. Inform. Comm. Eng., COMP97-25, (1997), pp. 25-32.
12. S. Kapoor, *On minimum 3-cuts and approximating k -cuts using cut trees*, Lecture Notes in Computer Science 1084, Springer-Verlag, (1996), pp. 132-146.
13. D. R. Karger, *Minimum cuts in near-linear time*, Proceedings 28th ACM Symposium on Theory of Computing, (1996), pp. 56-63.
14. D. R. Karger and C. Stein, *An $\tilde{O}(n^2)$ algorithm for minimum cuts*, Proceedings 25th ACM Symposium on Theory of Computing, (1993), pp. 757-765.
15. D. R. Karger and C. Stein, *A new approach to the minimum cut problems*, J. ACM, vol.43, no.4, (1996), pp. 601-640.
16. A. V. Karzanov, *O nakhozhdennii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh*, In Matematicheskie Voprosy Upravleniya Proizvodstvom, volume 5, Moscow State University Press, Moscow, (1973). In Russian; title translation: On finding maximum flows in networks with special structure and some applications.
17. N. Katoh, T. Ibaraki and H. Mine, *An efficient algorithm for K shortest simple paths*, Networks, vol.12, (1982), pp. 441-427.
18. R. Klimmek and F. Wagner, *A simple hypergraph min cut algorithm*, Technical Report B 96-02, Department of Mathematics and Computer Science, Freie Universität Berlin (1996).
19. E. L. Lawler, *Cutsets and partitions of hypergraphs*, Networks, vol.3, (1973), pp. 275-285.
20. C. H. Lee, M. Kim and C. I. Park, *An efficient k -way graph partitioning algorithm for task allocation in parallel computing systems*, Proc. IEEE Int. Conf. on Computer-Aided Design, (1990), pp. 748-751.
21. T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley (1990).
22. H. Nagamochi and T. Ibaraki, *Computing the edge-connectivity of multigraphs and capacitated graphs*, SIAM J. Discrete Mathematics, vol.5, (1992), pp. 54-66.
23. H. Nagamochi and T. Ibaraki, *A note on minimizing submodular functions*, Information Processing Letters, vol. 67, (1998), pp.239-244.
24. H. Nagamochi, K. Nishimura and T. Ibaraki, *Computing all small cuts in undirected networks*, SIAM J. Discrete Mathematics, vol. 10, (1997), pp. 469-481.
25. H. Nagamochi, S. Katayama and T. Ibaraki, *A faster algorithm for computing minimum 5-way and 6-way cuts in graphs*, Technical Report, Department of Applied Mathematics and Physics, Kyoto University, 1999.
26. Pulleyblank, *Presentation at SIAM Meeting on Optimization*, MIT, Boston, MA (1982).
27. M. Queyranne, *Minimizing symmetric submodular functions*, Mathematical Programming, 82 (1998), pp. 3-12.
28. V. V. Vazirani and M. Yannakakis, *Suboptimal cuts: Their enumeration, weight, and number*, Lecture Notes in Computer Science, 623, Springer-Verlag, (1992), pp. 366-377.

Scheduling Two Machines with Release Times

John Noga¹ and Steve Seiden²

¹ Technische Universität Graz
Institut für Mathematik B
Steyrergasse 30/ 2. Stock
A-8010 Graz, Austria

² Max-Planck-Institute für Informatik
Im Stadtwald
D-66123 Saarbrücken, Germany

Abstract. We present a deterministic online algorithm for scheduling two parallel machines when jobs arrive over time and show that it is $(5 - \sqrt{5})/2 \approx 1.38198$ -competitive. The best previously known algorithm is $\frac{3}{2}$ -competitive. Our upper bound matches a previously known lower bound, and thus our algorithm has the best possible competitive ratio. We also present a lower bound of 1.21207 on the competitive ratio of any randomized online algorithm for any number of machines. This improves a previous result of $4 - 2\sqrt{2} \approx 1.17157$.

1 Introduction

We consider one of the most basic scheduling problems: scheduling parallel machines when jobs arrive over time with the objective of minimizing the makespan. This problem is formulated as follows: There are m machines and n jobs. Each job has a *release time* r_j and a *processing time* p_j . An algorithm must assign each job to a machine and fix a start time. No machine can run more than one job at a time and no job may start prior to being released. For job j let s_j be the *start time* in the algorithm's schedule. We define the *completion time* for job j to be $c_j = s_j + p_j$. The *makespan* is $\max_j c_j$. The algorithm's goal is to minimize the makespan.

We study an *online* version of this problem, where jobs are completely unknown until their release times. In contrast, in the *offline* version all jobs are known in advance. Since it is in general impossible to solve the problem optimally online, we consider algorithms which approximate the best possible solution.

Competitive analysis is a type of worst case analysis where the performance of an online algorithm is compared to that of the optimal offline algorithm. This approach to analyzing online problems was initiated by Sleator and Tarjan, who used it to analyze the List Update problem [17]. The term competitive analysis originated in [12]. For a given job set σ , let $\text{cost}_A(\sigma)$ be the cost incurred by an algorithm A on σ . Let $\text{cost}(\sigma)$ be the cost of the optimal schedule for σ . A scheduling algorithm A is ρ -competitive if

$$\text{cost}_A(\sigma) \leq \rho \cdot \text{cost}(\sigma),$$

for all job sets σ . The *competitive ratio* of $\mathcal{A}$ is the infimum of the set of values ρ for which $\mathcal{A}$ is ρ -competitive. Our goal is to find the algorithm with smallest possible competitive ratio.

A great deal of work has been done on parallel machine scheduling when jobs are presented in a list and must be immediately scheduled prior to any processing [1, 2, 3, 4, 5, 7, 8, 9, 11, 14, 15]. On the other hand, very little work has been done on the version which we address here, despite the fact that this version of the problem seems more realistic. The known results are as follows: Hall and Shmoys show that the LIST algorithm of Graham is 2-competitive for all m [9, 10]. In [16], Shmoys, Wein and Williamson present a general online algorithm for scheduling with release dates. This algorithm uses as a subroutine an offline algorithm for the given problem. If the offline algorithm is an α -approximation, then the online algorithm is 2α -competitive. They also show a lower bound of $\frac{10}{9}$ on the competitive ratio for online scheduling of parallel machines with release times. The LPT algorithm starts the job with largest processing time whenever a machine becomes available. Chen and Vestjens show that LPT is $\frac{3}{2}$ competitive for all m [6]. These same authors also show a lower bound of $3 - \varphi \approx 1.38197$ for $m = 2$ and a general lower bound of 1.34729, where $\varphi = (1 + \sqrt{5})/2 \approx 1.61803$ is the golden ratio. Stougie and Vestjens [18] show a randomized lower bound of $4 - 2\sqrt{2} \approx 1.17157$ which is valid for all m .

We present an algorithm for scheduling two parallel machines, called SLEEPY, and show that it is $(3 - \varphi)$ -competitive, and thus has the best possible competitive ratio. We also show a randomized lower bound of

$$1 + \max_{0 \leq u < 1} \frac{u}{1 - 2 \ln(1 - u)} > 1.21207.$$

Finally, we show that any distribution over two deterministic algorithms is at best $\frac{5}{4}$ -competitive. It is our hope that these results arouse interest in the general m -machine problem.

2 The SLEEPY Algorithm

We say that a job is *available* if it has been released, but not yet scheduled. A machine that is not processing a job is called *idle*. Define

$$\alpha = 2 - \varphi = \frac{3 - \sqrt{5}}{2} \approx 0.38197.$$

Our algorithm, which we call SLEEPY, is quite simple. If both machines are processing jobs or there are no available jobs then we have no choice but to wait. So, assume that there is at least one idle machine and at least one available job. If both machines are idle then start the available job with largest processing time. If at time t , one machine is idle and the other machine is running job j then start the available job with largest processing time if and only if $t \geq s_j + \alpha \cdot p_j$.

With this algorithm a machine can be idle for two reasons. If a machine is idle because $t < s_j + \alpha \cdot p_j$ we say the machine is *sleeping*. If a machine is idle because

there are no available jobs we say that the machine is *waiting*. If a machine is waiting we call the next job released *tardy*.

We claim that SLEEPY is $(1+\alpha)$ -competitive. By considering two jobs of size 1 released at time 0 we see that SLEEPY is no better than $(1+\alpha)$ -competitive.

3 Analysis

We will show, by contradiction, that no set of jobs with optimal offline makespan 1 causes SLEEPY to have a makespan more than $1+\alpha$. This suffices, since we can rescale any nonempty job set to have optimal offline makespan 1.

Without loss of generality, we identify jobs with integers $1, \dots, n$ so that $c_1 \leq c_2 \leq \dots \leq c_n$. Assume that $\varrho = \{(p_i, r_i)\}_{i=1}^n$ is a minimum size set of jobs with optimal offline makespan 1 which causes SLEEPY to have a makespan $c_n > 1+\alpha$. Let A be the machine which runs job n and B be the other machine.

The remainder of the analysis consists of verifying a number of claims. Intuitively, what we want to show is: 1) the last three jobs to be released are the last three jobs completed by SLEEPY, 2) two of the last three jobs have processing time greater than α and the third has processing time greater than $1-\alpha$, and 3) replacing the last three jobs with one job results in a smaller counterexample.

Claim (Chen and Vestjens [6]). Without loss of generality, there is no time period during which SLEEPY has both machines idle.

Proof. Assume both machines are idle during (t_1, t_2) and that there is no time later than t_2 when both machines are idle. If there is a job with release time prior to t_2 then removing this job results in a set with fewer jobs, the same optimal offline cost, and the same cost for SLEEPY. If no job is released before t_2 then $\varrho' = \{(p_i/(1-t_2), (r_i-t_2)/(1-t_2))\}_{i=1}^n$ is a job set with optimal makespan 1, greater cost for SLEEPY, and no time period when SLEEPY has both machines idle. Note ϱ' is simply ϱ with all release times decreased by t_2 and all jobs rescaled so that the optimal cost remains 1.

Claim. The last job on machine B completes at time $c_{n-1} < 1$.

Proof. Since there is no period when SLEEPY has both machines idle, the last job on machine B must complete at or after s_n and so $c_{n-1} \geq s_n$.

Let j be the last tardy job and k be the job running when the last tardy job is released. If there are no tardy jobs let $r_j = c_k = 0$. At time r_j the optimal algorithm may or may not have completed all jobs released before r_j . Let p (possibly 0) be the amount of processing remaining to be completed by the optimal algorithm at time r_j on jobs released before r_j .

Since ϱ is a minimum size counterexample, $c_k \leq (1+\alpha)(r_j+p)$. Note that between time r_j and c_n the processing done by SLEEPY is exactly the processing done by the optimal algorithm plus $c_k - r_j - p$.

It follows from the previous claim that there is a set of jobs $\sigma_1, \dots, \sigma_\ell$ such that $s_{\sigma_1} \leq c_k < c_{\sigma_1}$, $s_{\sigma_i} \leq s_{\sigma_{i+1}}$, and $c_{\sigma_\ell} = c_{n-1}$.

Consider the time period (c_k, c_{σ_1}) . During this period neither machine is waiting. So, at least $(2 - \alpha)(c_{\sigma_1} - c_k)$ processing is completed by SLEEPY. Similarly, during the time period $(c_{\sigma_i}, c_{\sigma_{i+1}})$ at least $(2 - \alpha)(c_{\sigma_{i+1}} - c_{\sigma_i})$ processing is completed by SLEEPY for $i = 1, 2, \dots, \ell - 1$. Therefore, during the period (c_k, c_{n-1}) , SLEEPY completes at least $(2 - \alpha)(c_{n-1} - c_k)$ processing.

Since the optimal algorithm can process at most $2(1 - r_j)$ after r_j ,

$$2(c_k - r_j) + (2 - \alpha)(c_{n-1} - c_k) + (c_n - c_{n-1}) - (c_k - r_j - p) \leq 2(1 - r_j).$$

Therefore,

$$(1 - \alpha)(c_{n-1} - c_k) \leq 2 - r_j - p - c_n.$$

and further

$$\begin{aligned} c_{n-1} &\leq \frac{2 - r_j - p - c_n}{1 - \alpha} + c_k \\ &= (2 - \alpha)(2 - r_j - p - c_n) + c_k \\ &< (2 - \alpha)(1 - \alpha - r_j - p) + c_k \\ &= 1 - (2 - \alpha)(r_j + p) + c_k \\ &\leq 1 - (2\alpha - 1)(r_j + p) \\ &\leq 1. \end{aligned}$$

The previous claim implies that $p_n \geq c_n - c_{n-1} > \alpha$.

Claim. $s_n > s_{n-1} + \alpha \cdot p_{n-1}$.

Proof. If $s_n \leq s_{n-1}$ then removing job $n - 1$ results in a smaller set with no greater offline cost and no less cost for SLEEPY, a contradiction. From the definition of SLEEPY, $s_n \geq s_{n-1} + \alpha \cdot p_{n-1}$. Suppose that this is satisfied with equality. If $p_n > p_{n-1}$ then $r_n \geq s_{n-1}$. Therefore, we have

$$c_n = s_n + p_n = s_{n-1} + \alpha \cdot p_{n-1} + p_n \leq \alpha \cdot p_{n-1} + r_n + p_n \leq 1 + \alpha.$$

On the other hand, if $p_n \leq p_{n-1}$ then

$$c_n = s_n + p_n \leq s_{n-1} + \alpha \cdot p_{n-1} + p_{n-1} \leq c_{n-1} + \alpha \cdot p_{n-1} \leq 1 + \alpha.$$

Claim. $c_{n-2} = s_n$.

Proof. Since $s_n > s_{n-1} + \alpha \cdot p_{n-1}$, either $c_{n-2} = s_n$ or machine A is idle immediately before time s_n . In the later case, we have $s_n = r_n$ and therefore $c_n = s_n + p_n = r_n + p_n \leq 1$, which is a contradiction.

Claim. $p_{n-1} \geq c_{n-2} - s_{n-1} > \alpha$.

Proof. It is easily seen that $p_{n-1} \geq c_{n-2} - s_{n-1}$. If $c_{n-2} - s_{n-1} \leq \alpha$ then $p_{n-1} = (c_{n-1} - c_{n-2}) + (c_{n-2} - s_{n-1}) \leq (c_{n-1} - c_{n-2}) + \alpha < p_n$. Therefore $p_{n-1} - c_{n-1} + c_{n-2} \leq \alpha$ and further $r_n \geq s_{n-1}$. This means $c_n = s_{n-1} + p_{n-1} + p_n - (c_{n-1} - c_{n-2}) \leq r_n + p_n + \alpha \leq 1 + \alpha$, a contradiction.

Claim. $p_{n-2} > \alpha$.

Proof. If $p_{n-2} \leq \alpha$ then $p_{n-2} < p_n$. So, $r_n \geq s_{n-2}$. This means $c_n = s_{n-2} + p_{n-2} + p_n \leq r_n + \alpha + p_n \leq 1 + \alpha$, a contradiction.

Of the jobs $n-1$ and $n-2$, let j be the job which starts first and k be the other.

Claim. $p_j > 1 - \alpha$.

Proof. We have $p_j = c_j - s_j \geq c_{n-2} - s_j$. Certainly, $s_j + \alpha \cdot p_j \leq s_k$. So, $(1 - \alpha)(c_{n-2} - s_j) \geq c_{n-2} - s_j - \alpha \cdot p_j \geq c_{n-2} - s_k$. Therefore,

$$\begin{aligned} p_j &\geq c_{n-2} - s_j \\ &\geq \frac{c_{n-2} - s_k}{1 - \alpha} \\ &= (2 - \alpha)(c_{n-2} - \max\{s_{n-1}, s_{n-2}\}) \\ &= (2 - \alpha) \min\{c_{n-2} - s_{n-1}, p_{n-2}\} \\ &> (2 - \alpha)\alpha \\ &= 1 - \alpha. \end{aligned}$$

Claim. $s_j + \alpha \cdot p_j = s_k$.

Proof. No job other than n , $n-1$, and $n-2$ requires more than α processing time. Note that jobs k and n must go on the same machine in the optimal schedule. This means either k or n must be released before time $1 - 2\alpha$. Suppose job i has $s_i \leq s_j$ and $c_i \geq s_j + \alpha \cdot p_j$. Then $s_i + \alpha \cdot p_i \leq s_j$ and $p_i \geq \alpha \cdot p_i + \alpha \cdot p_j \geq \alpha \cdot p_i + \alpha(1 - \alpha)$. This implies that $p_i \geq \alpha$, a contradiction. Therefore, any job that starts before job j must complete before time $s_j + \alpha \cdot p_j$: Since $\alpha \cdot p_j > \alpha(1 - \alpha) = 1 - 2\alpha$, we have $s_k = s_j + \alpha \cdot p_j$.

Claim. $\min\{r_n, r_{n-1}, r_{n-2}\} > \max_{i \leq n-3} \{s_i\}$.

Proof. Since jobs n , $n-1$, and $n-2$ are larger than all other jobs and start later, they must have been released after $\max_{i \leq n-3} \{s_i\}$.

Claim. $p_n + p_k - p_j \geq 0$.

Proof. We have

$$\begin{aligned} p_n + p_k - p_j &= p_n + c_k - s_k - (c_j - s_j) \\ &= p_n + c_k - c_j - \alpha \cdot p_j \\ &\geq p_n + c_k - c_j - \alpha \\ &= c_n - c_{n-1} + c_{n-1} - c_{n-2} + c_k - c_j - \alpha \\ &> c_{n-1} - c_{n-2} + c_k - c_j \\ &\geq 0. \end{aligned}$$

Let ϱ' be the set of jobs $\{(p_i, r_i)\}_{i=1}^{n-3} \cup \{(p_n + p_k - p_j, \min\{r_n, r_{n-1}, r_{n-2}\})\}$. This set is valid by the preceding claim.

Claim. The optimal cost of ϱ' is no more than $1 - p_j$.

Proof. Since we can assume that on each of the machines individually the optimal algorithm orders its jobs by release times, this set can be processed identically to ϱ for the first $n - 3$ jobs. The new job $(p_n + p_k - p_j, \min\{r_n, r_{n-1}, r_{n-2}\})$ runs on the machine which would have run k . This schedule has cost no more than $1 - p_j$.

Claim. SLEEPY's cost on ϱ' is at least $c_n - (1 + \alpha)p_j$.

Proof. The first $n - 3$ jobs are served identically. The final job starts at time s_j . The makespan is at least $s_j + p_n + p_k - p_j = s_k + p_k + p_n - (1 + \alpha)p_j \geq c_n - (1 + \alpha)p_j$.

If we rescale ϱ' to have optimal cost 1 then we have our contradiction (a smaller counterexample). We therefore have:

Theorem 1. SLEEPY is $(1 + \alpha)$ -competitive for $\alpha = 2 - \varphi$.

4 Randomized Lower Bounds

First we show a lower bound which improves on the previous result of Stougie and Vestjens [18, 19]:

Theorem 2. No randomized algorithm for $m \geq 2$ machines is ρ -competitive with ρ less than

$$1 + \max_{0 \leq u < 1} \frac{u}{1 - 2 \ln(1 - u)} > 1.21207.$$

Proof. We make use of the von Neumann/Yao principle [21, 20]. Let $0 \leq u < 1$ be a real constant. We show a distribution over job sets such that the expectation of the competitive ratio of all deterministic algorithms is at least $1 + u/(1 - 2 \ln(1 - u))$. Let

$$q = \frac{1}{1 - 2 \ln(1 - u)}, \quad p(y) = \frac{1}{(y - 1) \ln(1 - u)}.$$

We now define the distribution. In all cases, we give two jobs of size 1 at time 0. With probability q , we give no further jobs. With probability $1 - q$, we pick y at random from the interval $[0, u]$ using the probability density $p(y)$ and release $m - 1$ jobs of size $2 - y$ at time y . Note that $q \geq 0$ and that $\int_0^u p(y) dy = 1$, and therefore this is a valid distribution.

We now analyze the expected competitive ratio incurred by a deterministic online algorithm on this distribution. Let x be the time at which the algorithm would start the second of the two size one jobs, given that it never receives the jobs of size $2 - y$. If the third job is not given, the algorithm's cost is $1 + x$ while the optimal offline cost is 1. If the jobs of size $2 - y$ are given at time $y \leq x$, then the algorithm's cost is at least 2 and this is also the optimal offline cost. If the jobs of size $2 - y$ are given at time $y > x$, then the algorithm's cost is at

least $3 - y$ while the optimal offline cost is again 2. First consider $x \geq u$. In this case, the expected competitive ratio is

$$\begin{aligned} E_{\sigma} \left[\frac{\text{cost}_A(\sigma)}{\text{cost}(\sigma)} \right] &\geq q(1+x) + (1-q) \int_0^u p(y) dy \\ &\geq \frac{1+u}{1-2\ln(1-u)} + \frac{-2\ln(1-u)}{1-2\ln(1-u)} \int_0^u \frac{dy}{(y-1)\ln(1-u)} \\ &= 1 + \frac{u}{1-2\ln(1-u)}. \end{aligned}$$

Now consider $x < u$. In this case we have

$$\begin{aligned} E_{\sigma} \left[\frac{\text{cost}_A(\sigma)}{\text{cost}(\sigma)} \right] &\geq q(1+x) + (1-q) \left(\int_0^x p(y) dy + \int_x^u p(y) \frac{3-y}{2} dy \right) \\ &= \frac{1+x}{1-2\ln(1-u)} + \frac{-2}{1-2\ln(1-u)} \left(\int_0^x \frac{dy}{y-1} + \int_x^u \frac{3-y}{2} \frac{dy}{y-1} \right) \\ &= \frac{1+x}{1-2\ln(1-u)} + \frac{-2}{1-2\ln(1-u)} \left(\ln(u-1) - \frac{u}{2} + \frac{x}{2} \right) \\ &= 1 + \frac{u}{1-2\ln(1-u)}. \end{aligned}$$

Since the adversary may choose u , the theorem follows. Choosing $u = 0.575854$ yields a bound of at least 1.21207.

One natural class of randomized online algorithms is the class of *barely random* algorithms [13]. A barely random online algorithm is one which is distribution over a constant number of deterministic algorithms. Such algorithms are often easier to analyze than general randomized ones. We show a lower bound for any barely random algorithm which is a distribution over two deterministic algorithms:

Theorem 3. *No distribution over two deterministic online algorithms is ρ -competitive with ρ less than $\frac{5}{4}$.*

Proof. Suppose there is an algorithm which is $(\frac{5}{4} - \epsilon)$ -competitive for $\epsilon > 0$. We give the algorithm two jobs of size 1 at time 0. With some probability p , the algorithm starts the second job at time t_1 , while with probability $1 - p$, it starts it at time t_2 . Without loss of generality, $t_1 < t_2$. First, we must have

$$1 + pt_1 + (1-p)t_2 < \frac{5}{4},$$

and therefore

$$p > \frac{4t_2 - 1}{4(t_2 - t_1)}.$$

We also conclude that $t_1 < \frac{1}{4}$. Now suppose we give the algorithm $m - 1$ jobs of size $2 - t_1 - \delta$ at time $t_1 + \delta$. Then its competitive ratio is at least

$$\frac{p(3 - t_1 - \delta) + (1 - p)2}{2} = \frac{1 - t_1 - \delta}{2}p + 1 > \frac{(1 - t_1 - \delta)(4t_2 - 1)}{8(t_2 - t_1)} + 1 < \frac{5}{4},$$

and therefore

$$t_2 < \frac{1 - 3t_1 - \delta}{2 - 4t_1 - 4\delta}.$$

Since this is true for all $\delta > 0$ we have

$$t_2 < \frac{1 - 3t_1}{2 - 4t_1}.$$

If instead we give $m - 1$ jobs of size $2 - t_2 - \delta$ at time $t_2 + \delta$, then the competitive ratio is at least

$$\frac{3 - t_2 - \delta}{2} > \frac{5 - 9t_1 - 2\delta + 4t_1\delta}{4 - 8t_1}.$$

Again, this holds for all $\delta > 0$ and so we have

$$\frac{5 - 9t_1}{4 - 8t_1} \leq \frac{5}{4} - \epsilon,$$

which is impossible for $0 \leq t_1 < \frac{1}{4}$.

5 Conclusions

We have shown a deterministic online algorithm for scheduling two machines with release times with the best possible competitive ratio. Further, we have improved the lower bound on the competitive ratio of randomized online algorithms for this problem.

We feel there are a number of interesting and related open questions. First, is there a way to generalize SLEEPY to an arbitrary number of machines and have a competitive ratio smaller than LPT? Or the even stronger result, for a fixed value of $\epsilon > 0$ is there an algorithm which is $(3/2 - \epsilon)$ -competitive for all m ? Second, does randomization actually reduce the competitive ratio? It seems like randomization should help to reduce the competitive ratio a great deal. However, the best known randomized algorithms are actually deterministic (SLEEPY for $m = 2$ and LPT for $m > 2$). Third, how much does the competitive ratio decrease if restarts are allowed? In many real world situations a job can be killed and restarted later with only the loss of processing already completed.

6 Acknowledgement

This research has been supported by the START program Y43-MAT of the Austrian Ministry of Science. The authors would like to thank Gerhard Woeginger for suggesting the problem.

References

- [1] ALBERS, S. Better bounds for online scheduling. In *Proceedings of the 29th ACM Symposium on Theory of Computing* (1997), pp. 130–139.
- [2] BARTAL, Y., FIAT, A., KARLOFF, H., AND VOHRA, R. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences* 51, 3 (Dec 1995), 359–366.
- [3] BARTAL, Y., KARLOFF, H., AND RABANI, Y. A better lower bound for on-line scheduling. *Information Processing Letters* 50, 3 (May 1994), 113–116.
- [4] CHEN, B., VAN VLIET, A., AND WOEGINGER, G. A lower bound for randomized on-line scheduling algorithms. *Information Processing Letters* 51, 5 (Sep 1994), 219–222.
- [5] CHEN, B., VAN VLIET, A., AND WOEGINGER, G. New lower and upper bounds for on-line scheduling. *Operations Research Letters* 16, 4 (Nov 1994), 221–230.
- [6] CHEN, B., AND VESTJENS, A. Scheduling on identical machines: How good is LPT in an online setting? *Operations Research Letters* 21, 4 (Nov 1997), 165–169.
- [7] FAIGLE, U., KERN, W., AND TURÁN, G. On the performance of on-line algorithms for partition problems. *Acta Cybernetica* 9, 2 (1989), 107–119.
- [8] GALAMBOS, G., AND WOEGINGER, G. An online scheduling heuristic with better worst case ratio than Graham’s list scheduling. *SIAM Journal on Computing* 22, 2 (Apr 1993), 349–355.
- [9] GRAHAM, R. L. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal* 45 (1966), 1563–1581.
- [10] HALL, L., AND SHMOYS, D. Approximation schemes for constrained scheduling problems. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science* (1989), pp. 134–139.
- [11] KARGER, D., PHILLIPS, S., AND TORNG, E. A better algorithm for an ancient scheduling problem. *Journal of Algorithms* 20, 2 (Mar 1996), 400–430.
- [12] KARLIN, A., MANASSE, M., RUDOLPH, L., AND SLEATOR, D. Competitive snoopy caching. *Algorithmica* 3, 1 (1988), 79–119.
- [13] REINGOLD, N., WESTBROOK, J., AND SLEATOR, D. Randomized competitive algorithms for the list update problem. *Algorithmica* 11, 1 (Jan 1994), 15–32.
- [14] SEIDEN, S. S. Randomized algorithms for that ancient scheduling problem. In *Proceedings of the 5th Workshop on Algorithms and Data Structures* (Aug 1997), pp. 210–223.
- [15] SGALL, J. A lower bound for randomized on-line multiprocessor scheduling. *Information Processing Letters* 63, 1 (Jul 1997), 51–55.
- [16] SHMOYS, D., WEIN, J., AND WILLIAMSON, D. Scheduling parallel machines on-line. In *Proceedings of the 32nd Symposium on Foundations of Computer Science* (Oct 1991), pp. 131–140.
- [17] SLEATOR, D., AND TARJAN, R. Amortized efficiency of list update and paging rules. *Communications of the ACM* 28, 2 (Feb 1985), 202–208.
- [18] STOUGIE, L., AND VESTJENS, A. P. A. Randomized on-line scheduling: How low can’t you go? Manuscript, 1997.
- [19] VESTJENS, A. P. A. *On-line Machine Scheduling*. PhD thesis, Eindhoven University of Technology, The Netherlands, 1997.
- [20] VON NEUMANN, J., AND MORGENSTERN, O. *Theory of games and economic behavior*, 1st ed. Princeton University Press, 1944.
- [21] YAO, A. C. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science* (1977), pp. 222–227.

An Introduction to Empty Lattice Simplices

András Sebő*

CNRS, Laboratoire Leibniz-IMAG, Grenoble, France

<http://www-leibniz.imag.fr/DMD/OPTICOMB>

Abstract. We study simplices whose vertices lie on a lattice and have no other lattice points. Such ‘empty lattice simplices’ come up in the theory of integer programming, and in some combinatorial problems. They have been investigated in various contexts and under varying terminology by Reeve, White, Scarf, Kannan and Lovász, Reznick, Kantor, Haase and Ziegler, etc.

Can the ‘emptiness’ of lattice simplices be ‘well-characterized’? Is their ‘lattice-width’ small? Do the integer points of the parallelepiped they generate have a particular structure?

The ‘good characterization’ of empty lattice simplices occurs to be open in general! We provide a polynomial algorithm for deciding when a given integer ‘knapsack’ or ‘partition’ lattice simplex is empty. More generally, we ask for a characterization of linear inequalities satisfied by the lattice points of a lattice parallelepiped. We state a conjecture about such inequalities, prove it for $n \leq 4$, and deduce several variants of classical results of Reeve, White and Scarf characterizing the emptiness of small dimensional lattice simplices. For instance, a three dimensional integer simplex is empty if and only if all its faces have width 1. Seemingly different characterizations can be easily proved from one another using the Hermite normal form.

In fixed dimension the width of polytopes can be computed in polynomial time (see the simple integer programming formulation of Haase and Ziegler). We prove that it is already NP-complete to decide whether the width of a very special class of integer simplices is 1, and we also provide for every $n \geq 3$ a simple example of n -dimensional empty integer simplices of width $n - 2$, improving on earlier bounds.

1 Introduction

Let $V \subseteq \mathbb{R}^n$ ($n \in \mathbb{N}$) be a finite set. A *polytope* is a set of the form $\text{conv}(V) := \{\sum_{v \in V} \lambda_v v : \sum_{v \in V} \lambda_v = 1\}$. If V is linearly independent, then $S := \text{conv}(V \cup \{0\})$ is called a *simplex*. (It will be assumed that $0 \in \mathbb{R}^n$ is one of the vertices of S .) The (linear or affine) *rank* $r(S)$ is the linear rank of V . A polytope is said to be *integer* if its vertices are integer vectors. More generally, fixing an arbitrary ‘lattice’ L , a lattice polytope is a polytope whose vertices are in L . The results we are proving in this paper do hold for arbitrary lattices, but this general case

* Research developed partly during a visit in the Research Institute for Mathematical Sciences, Kyoto University.

can always be obviously reduced to $L = \mathbb{Z}^n$. Therefore we will not care about more general lattices.

A set of the form $\text{cone}(V) := \{\sum_{v \in V} \lambda_v v : \lambda_v \geq 0\}$ is a *cone*; if V is linearly independent, the cone is called *simplicial*.

We refer to Schrijver [13] for basic facts about polytopes, cones and other notions of polyhedral combinatorics, as well as for standard notations such as the affine or linear hull of vectors, etc.

Let us call an integer polytope P *empty*, if it is integer, and denoting the set of its vertices by V , $(P \cap \mathbb{Z}^n) \setminus V = \emptyset$. (The definition is similar for arbitrary lattice L instead of $\mathbb{Z}^n$.) Empty lattice polytopes have been studied in the past four decades, let us mention as landmarks Reeve [11], White [15], Reznick [10], Scarf [12] and Haase, Ziegler [5]. (However, there is no unified terminology about them: the terms range from ‘lattice-point-free lattice polytopes’ to ‘elementary or fundamental lattice polytopes’.) The latter paper is devoted to the volume and the width of empty lattice simplices.

In this paper we study the structure of empty lattice simplices, the correlation of emptiness and the width of lattice simplices, including the computational complexity of both. (Note that deciding whether a (not necessarily integer) simplex contains an integer point is trivially NP-complete, since the set of feasible solutions of the knapsack problem $K := \{x \in \mathbb{R}^n : ax = b, x \geq 0\}$ ($a \in \mathbb{N}^n, b \in \mathbb{N}$) is a simplex. However, in Section 2 we will decide whether a knapsack lattice simplex is empty in polynomial time.)

Deciding whether a lattice simplex is empty is the simplest of a possible range of problems that can be formulated as follows: given linearly independent integer vectors $a_1, \dots, a_n \in \mathbb{Z}^n$ is there an integer vector v in the cone they generate such that the uniquely determined coefficients $\lambda_1, \dots, \lambda_n \in \mathbb{Q}_+$ for which $\lambda_1 a_1 + \dots + \lambda_n a_n = v$ satisfy some given linear inequalities. The problem investigated here corresponds to the inequality $\lambda_1 + \dots + \lambda_n \leq 1$. Another interesting variant is the existence of an integer point where the λ_i ($i = 1, \dots, n$) satisfy some lower and upper bounds. For instance the ‘Lonely Runner Problem’ (or ‘View Obstruction Problem’) for velocity vector $v \in \mathbb{R}^n$ (see [4]) can be restated as the problem of the existence of an integer vector with $1/(n+1) \leq \lambda_i \leq n/(n+1)$ for all $i = 1, \dots, n$ in $\text{cone}(e_1, \dots, e_n, (v, d))$, where the e_i ($i = 1, \dots, n$) are unit vectors and d is the least common multiple of all the vectors $v_i + v_j$, ($i, j = 1, \dots, n$).

The *width* $W(S)$ in $\mathbb{R}^n$ of a set $S \subseteq \mathbb{R}^n$ is the minimum of $\max\{w^T(x - y) : x, y \in S\}$ over all vectors $w \in \mathbb{Z}^n \setminus \{0\}$. If the rank of S is $r < n$, then the width of S in $\mathbb{R}^n$ is 0, and it is more interesting to speak about its *width in* $\text{aff}(S) = \text{lin}(S)$ defined as the minimum of $\max\{w^T(x - y) : x, y \in S\}$ over all vectors $w \in \mathbb{Z}^n$ not orthogonal to $\text{lin}(S)$. Shortly, the *width* of S will mean its width in $\text{aff}(S)$.

If $0 \notin V \subseteq \mathbb{Z}^n$, and V is linearly independent, then define $\text{par}(V) := \{x \in \mathbb{Z}^n : x = \sum_{v \in V} \lambda_v v; 1 > \lambda_v \geq 0 (v \in V)\}$ and call it a *parallelepiped*. If in addition $|V| = n$, then $|\text{par}(V)| = \det(V)$, where $\det(V)$ denotes the *absolute value of the determinant* of the matrix whose rows are the elements of V (see

for instance Cassels [3], or for an elementary proof see Sebő [14]). In particular, $\text{par}(V) = \{0\}$ if and only if $\det(V) = 1$.

The problem of deciding the emptiness of parallelepipeds generated by integer vectors is therefore easy. On the other hand the same problem is still open for simplices, and that is the topic of this paper.

If $n \leq 3$, it is well-known [15], that the width of an empty integer simplex is 1 (the reverse is even easier – for both directions see Corollary 4.5 below).

The following example shows that in general the width of empty integer simplices can be large. The problem might have been overlooked before: the simple construction below is apparently the first explicit example of empty integer simplices of arbitrary high width. The best result known so far was Kantor's non-constructive proof [8] for the existence of integer simplices of width n/e . For a survey concerning previous results on the correlation of the width and the volume of empty lattice simplices see Haase, Ziegler [5].

It is easy to construct integer simplices of width n without integer points at all (not even the vertices), for instance by the following well-known example, [6]: for arbitrary $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$, the simplex $\{x \in \mathbb{R}^n : \sum_{i=1}^n x_i \leq n - \varepsilon/2, x_i \geq \varepsilon/2 \text{ for all } i = 1, \dots, n\}$ has width $n - \varepsilon$ and no integer point. The vertices of this simplex have exactly one big coordinate. We define an integer simplex which is 'closest possible' to this one:

Let $k \in \mathbb{N}$ (the best choice will be $k = n - 2$). $S_n(k) := \text{conv}(s_0, s_1, \dots, s_n)$, $s_0 := 0$, $s_1 := (1, k, 0, \dots, 0)$, $s_2 := (0, 1, k, 0, \dots, 0)$, $\dots$, $s_{n-1} := (0, \dots, 0, 1, k)$, $s_n := (k, 0, \dots, 0, 1)$. Let $s_{i,j}$ denote the j -th coordinate of s_i , that is, $s_{i,j} = 0$ if $|i - j| \geq 2$, $s_{i,i} = 1$, $s_{i,i+1} = k$, ($i, j \in \{1, \dots, n\}$). The notation $i + 1$ is understood mod n .

(1.1) *The width of $S_n(k)$ is k , unless both $k = 1$ and n is even.*

Indeed, since $0 \in S_n(k)$, the width is at least k if and only if for arbitrary nonzero integer n -dimensional w there exists $i \in \{1, \dots, n\}$ so that $|w^T s_i| \geq k$. The polytope $S_n(k)$ is a simplex if $s_1, \dots, s_n$ are linearly independent, which is automatic if $W(S_n(k)) = k > 0$ holds.

Let $w \in \mathbb{Z}^n$, $w \neq 0$. If $w_i = 0$ for some $i \in \{1, \dots, n\}$, then there also exists one such that $w_i = 0$ and $w_{i+1} \neq 0$. But then $|w^T s_i| \geq s_{i,i+1} = k$, and we are done. So suppose w has full support. If there exists an $i = 1, \dots, n$ such that $|w_i| < |w_{i+1}|$, then $|w^T s_i| = |w_i + w_{i+1}k| \geq |w_{i+1}k| - |w_i| > (k - 1)|w_i| \geq k - 1$, and we are done again.

Hence, we can suppose that $|w_i| \geq |w_{i+1}|$ holds for all $i = 1, \dots, n$. But then the equality follows throughout, and we can suppose $|w_i| = 1$ for all $i = 1, \dots, n$. If there exists an $i \in \{1, \dots, n\}$ so that w_i, w_{i+1} have the same sign, then $w^T s_i = k + 1$. If the signs of the w_i are cyclically alternating, then $w^T s_i$ is also alternating between $(k - 1)$ and $-(k - 1)$ and then $w^T(s_i - s_{i+1}) = 2k - 2$. In this case n is even, and $2k - 2 \geq k$ unless $k = 1$.

So the width is at least k , unless n is even, and $k = 1$. Choosing w to be any of the n unit vectors we see that the width of $S_n(k)$ is k . $\square$

(1.2) If $k + 1 < n$, then $S_n(k)$ is an empty integer simplex.

Indeed, for a contradiction, let $z = (z_1, \dots, z_n) \in S_n(k)$ be an integer vector different from s_i ($i = 1, \dots, n$), $z = \sum_{i=0}^n \lambda_i s_i$, $\lambda_i \in \mathbb{R}_+$, $\sum_{i=0}^n \lambda_i = 1$.

Claim: $\lambda_i > 0$ for all $i = 1, \dots, n$.

Indeed, if not, there exists an $i \in \{1, \dots, n\}$ so that $\lambda_i = 0$, $\lambda_{i+1} \neq 0$. Then $z_{i+1} = \lambda_{i+1}$. Since $\lambda_{i+1} > 0$ by assumption, and $\lambda_{i+1} < 1$ since z is different from s_{i+1} , we have: z_{i+1} is not integer, contradicting the integrality of z . The claim is proved.

The claim immediately implies $z > 0$ and hence $z \geq 1$. Therefore $\sum_{i=1}^n z_i \geq n$. On the other hand, for all the vertices x of $S_n(k)$, $\sum_{i=1}^n x_i \leq k + 1$. Thus, if $k + 1 < n$, then $z \notin S_n(k)$. $\square$

Hence, for $n \geq 3$, $S_n(n-2)$ is an integer simplex of width $n-2$ by (1.1), and is empty by (1.2). The volume (determinant) of $S_n(n-2)$ is also high among empty simplices in $\mathbb{R}^n$. This example is not best possible : for small n there exist empty integer simplices of larger width, see [5]. Moreover Imre Bárány pointed out that in the above example, for odd n , the facet of $S_n(k)$ not containing 0 has still the same width as $S_n(k)$, providing an empty integer simplex of width $n-1$ if $n \geq 4$ is even; Bárány also noticed that the volume can be increased by a constant factor by changing one entry in the example. However, it still seems to be reasonable to think that the maximum width of an empty integer simplex $S \subseteq \mathbb{R}^n$ is $n + \text{constant}$. Kannan and Lovász [6] implies that the width of an arbitrary empty integer polytope is at most $O(n^2)$; in [1] this is improved to $O(n^{3/2})$, where for empty integer simplices $O(n \log n)$ is also proved.

The main question we are interested in is the following:

Question: Is there a simple ‘good-characterization’ theorem or even a polynomial algorithm deciding the emptiness of integer simplices ? How are the emptiness and the width correlated ? Is there any relation between their computational complexity ?

It is surprising that the literature does not make any claim about these problems in general. In the present paper we state some questions and provide some simple answers whenever we can. Unfortunately we do not yet know what is the complexity of deciding the emptiness of integer simplices, nor the complexity of computing the width of empty integer simplices !

In Section 2 we describe a good-characterization theorem and polynomial algorithm deciding if an integer knapsack (or ‘partition’) polytope is empty.

In Section 3 and Section 4 we show a possible good certificate for certain lattice simplices to be empty. As a consequence, a simple new proof is provided for facts well-known from [15], [12].

Deciding whether a lattice polytope is empty can be trivially reduced to the existence of an integer point in a slightly perturbed polytope. The result of Section 2 suggests that a reduction in the opposite direction could be more difficult, and is impossible in some particular cases.

This reduction implies that in fixed dimension it can be decided in polynomial time whether a lattice polytope is empty by Lenstra [9]. More generally,

Barvinok [2] developed a polynomial algorithm for counting the number of integer points in polytopes when the dimension is fixed.

We do not care about how the simplices are given, since the constraints can be computed from the vertices in polynomial time, and vice versa.

In Section 5 we explore the complexity of computing the width of simplices.

2 Knapsack Simplices

If $a, b \in \mathbb{Z}$, $\text{lcm}(a, b)$ denotes the least common multiple of a and b , that is, the smallest nonnegative number which is both a multiple of a and of b .

Theorem 2.1 *Let $K := \{x \in \mathbb{R}^n : a^T x = b, x \geq 0\}$, ($a \in \mathbb{N}^n, b \in \mathbb{N}$) be an integer polytope. Then K is empty, if and only if $\text{lcm}(a_i, a_j) = b$ for all $i \neq j = 1, \dots, n$.*

Proof. Clearly, K is an $n - 1$ -dimensional simplex whose vertices are $v_i := k_i e_i$, where $k_i := b/a_i$. Since K is an integer polytope, $k_i \in \mathbb{Z}$, that is, $a_i | b$ ($i = 1, \dots, n$).

Let us realize that K contains no integer points besides the vertices, if and only if $\gcd(k_i, k_j) = 1$ for all $i, j = 1, \dots, n$.

Indeed, if say $\gcd(k_1, k_2) = d > 1$ then $(1/d)v_1 + ((d-1)/d)v_2$ is an integer point. Conversely, suppose that there exists an integer vector $w = \sum_{i=1}^n \lambda_i v_i$ ($\lambda_i \geq 0, i = 1, \dots, n$), $\sum_{i=1}^n \lambda_i = 1$, and say $\lambda_1 > 0$. Let $\lambda_i := p_i/q_i$, where p_i, q_i are relatively prime nonzero integers whenever $\lambda_i \neq 0$. Since $\lambda_2 + \dots + \lambda_n = 1 - \lambda_1$, an arbitrary prime factor p of q_1 occurs also in q_i for some $i \in \{2, \dots, n\}$. Since $w_i = \lambda_i k_i$ is integer, the denominator of λ_i divides k_i , that is, $p | k_1, p | k_i$, proving $\gcd(k_1, k_i) \geq p > 1$.

Now it only remains to notice that for any $i, j = 1, \dots, n$, $\gcd(k_i, k_j) = 1$ if and only if $\text{lcm}(a_i, a_j) = \text{lcm}(b/k_i, b/k_j) = b$. □

This assertion does not generalize, the simplex K is quite special :

Corollary 2.2 *The integer knapsack polytope $K := \{x \in \mathbb{R}^n : a^T x = b, x \geq 0\}$, ($a \in \mathbb{N}^n, b \in \mathbb{N}$) is empty, if and only if all its two dimensional faces are empty.*

3 Parallelepiped Structure and Jump Coefficients

In this section we are stating two lemmas that will be needed in the sequel. The first collects some facts about the structure of parallelepipeds. The second is a result of number theoretic character. The structure provided by the first raises the problem solved by the second.

A *unimodular transformation* is a linear transformation defined by an integer matrix whose determinant is 1. An equivalent definition: a unimodular transformation is the composition of a finite number of reflections $f_i(x) := (x_1, \dots, -x_i, \dots, x_n)$, and sums $f_{i,j} := (x_1, \dots, x_{i-1}, x_i + x_j, x_{i+1}, \dots, x_n)$,

$(i, j = 1, \dots, n)$. The equivalence of the two definitions is easy to prove in knowledge of the Hermite normal form (see the definition in [13]).

We will use unimodular transformations of a set of vectors V by putting them into a matrix M as rows, and then using column operations to determine the Hermite normal form M' of M . Then the rows of M' can be considered to provide an ‘isomorphic’ representation of V .

The residue of $x \in \mathbb{R} \bmod d \in \mathbb{N}$ will be denoted by $\text{mod}(x, d)$, $0 \leq \text{mod}(x, d) < d$.

Let $V := \{v_1, \dots, v_n\} \subseteq \mathbb{Z}^n$ be a basis of $\mathbb{R}^n$, and $d := \det(v_1, \dots, v_n)$. By Cramer’s rule every integer vector is a linear combination of V with coefficients that are integer multiples of $1/d$. For $x \in \mathbb{R}^n$ the coefficient vector $\lambda = (\lambda_1, \dots, \lambda_n) \in \mathbb{Z}^n$ defined by the unique combination $x = (\lambda_1 v_1 + \dots + \lambda_n v_n)/d$, will be called the V -coefficient vector of x . In other words $\lambda = dV^{-1}x$ (where V denotes the $n \times n$ matrix whose n -th column is v_n). If $x \in \mathbb{Z}^n$ then all V -coefficients of x are integer.

Clearly, V -coefficient vectors are unchanged by linear transformations, and the width is unchanged under unimodular transformations. (The inverse of a unimodular transformation is also unimodular. Unimodular transformations can be considered to be the ‘isomorphisms’ of polytopes with respect to their integer vectors.)

A $\text{par}(V)$ -coefficient vector is a vector $\lambda \in \mathbb{R}^n$ which is the V -coefficient vector of some $x \in \text{par}(V)$. We will often exploit the fact that parallelepipeds are *symmetric* objects: if $x \in \text{par}(v_1, \dots, v_n)$, then $v_1 + \dots + v_n - x \in \text{par}(v_1, \dots, v_n)$. In other words, if $(\lambda_1, \dots, \lambda_n) \in \mathbb{Z}^n$ is a $\text{par}(V)$ -coefficient vector, then $(d - \lambda_1, \dots, d - \lambda_n)$ is also one (extensively used in [10] and [14]).

We will use $\text{par}(V)$ -coefficients and some basic facts in the same way as we did in [14]. These have been similarly used in books, for $n = 3$ in White’s paper [15], or in Reznick [10] through ‘barycentric coordinates’, whose similarity was kindly pointed out to the author by Jean-Michel Kantor.

An important part of the literature is in fact involved more generally with the number of integer points in three (or higher) dimensional simplices. Our ultimate goal is to find only simple general ‘good certificates’ (or polynomial algorithms) for deciding when this number is *zero*, and $\text{par}(V)$ -coefficients seem to be helpful to achieve this task. We are able to achieve only much less: we treat small dimensional cases in a simple new way, bringing to the surface some more general facts and conjectures.

If $x \in \mathbb{R}^n$, then clearly, there exists a unique integer vector p so that $x + p \in \text{par}(V)$. If $x \in \mathbb{Z}^n$ and $\lambda \in \mathbb{Z}^n$ is the V -coefficient vector of x , then the V -coefficient vector of $x + p$ is $(\text{mod}(\lambda_1, d)/d, \dots, \text{mod}(\lambda_n, d)/d)$. The $\text{par}(V)$ -coefficient vectors form a group $G = G(V)$ with respect to $\text{mod } d$ addition. This is the factor-group of the additive group of integer vectors with respect to the subgroup generated by V , moreover the following well-known facts hold:

Lemma 3.1 *Let $V := v_1, \dots, v_n \in \mathbb{Z}^n$. Then:*

- (a) $\text{par}(V \setminus v_n) = \{0\}$ if and only if there exists a unimodular transformation (and possibly permutation of the coordinates) such that $v_i := e_i$ ($i = 1, \dots, n - 1$),

$v_n = (a_1, \dots, a_{n-1}, d)$, where $d = \det(v_1, \dots, v_n)$ and $0 < a_i < d$ for all $i = 1, \dots, n-1$.

(b) If $\text{par}(V \setminus v_n) = \{0\}$, then $G(V)$ is a cyclic group.

(c) If $\text{par}(V \setminus v_n) = \{0\}$ then $\text{par}(V \setminus v_i) = \{0\}$ for some $i \in \{1, \dots, n-1\}$ if and only if in (a) $\gcd(a_i, d) = 1$.

Indeed, let the rows of a matrix M be the vectors $v_i \in \mathbb{Z}^n$ ($i = 1, \dots, n$) and consider the Hermite normal form of (the column lattice of) M . If $\text{par}(V \setminus v_n) = \{0\}$, then deleting the last row and column we get an $(n-1) \times (n-1)$ identity matrix, and (a), follows. Now (b) is easy, since $\text{par}(V) = \{0\} \cup \{[i/d(a_1, \dots, a_{n-1}, d)] : i = 1, \dots, d-1\}$, that is, the $\text{par}(V)$ -coefficients are equal to $\{\text{mod}(ih, d) : i = 1, \dots, d-1\}$, where $h := (d - a_1, \dots, d - a_{n-1}, 1)$. Statement (c) is also easy, since $\gcd(a_i, d) = \gcd(d - a_i, d) > 1$ if and only if there exists $j \in \mathbb{N}$, $j < d$ so that the i -th coordinate of $\text{mod}(jh, d)$ is 0.

Statement (b) is very useful for proving properties for all the parallelepiped P : one can generate the $V := \{v_1, \dots, v_n\}$ -coefficients of all the $d-1$ nonzero points of P by taking the mod d multiples of the V -coefficient vector of some generator $h = (h_1, \dots, h_n) \in P$ (cf. [10], or [14]). If the polytope we are considering is described in terms of inequalities satisfied by a function of the $\text{par}(V)$ -coefficients, then it is useful to understand how the $\text{par}(V)$ -coefficient vector $\text{mod}((i+1)h, d)$ changed comparing to $\text{mod}(ih, d)$.

For instance, for the simplex $S := \text{conv}(v_1, \dots, v_n)$ to be empty means exactly that the sum of the V -coefficients of any vector in P is strictly greater than d . For any coordinate $0 < a \leq d-1$ of h one simply has $\text{mod}((i+1)a, d) = \text{mod}(ia, d) + a$, unless the interval $(\text{mod}(ia, d), \text{mod}(ia, d) + a]$ contains a multiple of d , that is, if and only if $\text{mod}(ia, d) + a \geq d$. In this latter case $\text{mod}((i+1)a, d) = \text{mod}(ia, d) + a - d$, and we will say that i is a *jump-coefficient* of $a \bmod d$.

Hence mod d inequalities can be treated as ordinary inequalities, corrected by controlling of jump-coefficients. We will need only the following simply stated Lemma 3.2, relating containment relations between the sets of jump coefficients, to divisibility relations.

We say that $i \in \{1, \dots, d-1\}$ is a *jump-coefficient* of $a \in \mathbb{N} \bmod d$ ($1 \leq a < d$), if $\lfloor (i+1)a/d \rfloor > \lfloor ia/d \rfloor$ (equivalently, if $\text{mod}((i+1)a, d) < \text{mod}(ia, d)$). If $a = 1$, then $J_a(d) = \emptyset$, and if $a \geq 2$, the set of jump-coefficients of $a \bmod d$ is

$$(*) \quad J_a(d) := \{\lfloor id/a \rfloor : i = 1, \dots, a-1\}.$$

Let us illustrate our goal and the statement we want to prove on the easy example of $a = 2$ and d odd: let us show that $J_a(d) \subseteq J_b(d)$ if and only if b is even. Indeed, 2 has just one jump-coefficient, $\lfloor d/2 \rfloor$. So $J_a(d) \subseteq J_b(d)$ if and only if $\lfloor d/2 \rfloor \in J_b$, that is, if and only if the interval $(\lfloor d/2 \rfloor b, (\lfloor d/2 \rfloor + 1)b]$ contains a multiple of d . It does contain a multiple of $d/2$: $bd/2$, and since $b/2 < d/2$ this is the only multiple of $d/2$ it contains. But $bd/2$ is a multiple of d if and only if b is even, as claimed.

Lemma 3.2 states a generalization of this statement for arbitrary a . Let us first visualise the statement, and $(*)$ – a basic tool in the proof :

Let $d \in \mathbb{N}$ be arbitrary, and $0 < a, b < d$. The points $A := \{id/a : i = 1, \dots, a-1\}$ divide the interval $[0, d]$ into a equal parts. Each of these parts has

length bigger than 1, so the points of A lie in different intervals $(i, i + 1]$. Now (*) means exactly that

(**) $i \in J_a$ if and only if the interval $(i, i + 1]$ contains an element of A .

If $b > a$, then clearly, there is an interval $(i, i + 1]$ containing a point of B and not containing a point of A . If b is a multiple of a , then obviously, $B \supseteq A$. If $d - a$ is a multiple of $d - b$, then again $B \supseteq A$ is easy to prove (see the Fact below).

If $a, b \leq d/2$, then $d - b \mid d - a$ cannot hold. The following lemma states that under this condition the above remark can be reversed: if a does not divide b , then $A \setminus B \neq \emptyset$. If $a \leq d/2$ and $b > d/2$, then $J_a(d) \subseteq J_b(d)$ can hold without any of $a \mid b$ or $d - b \mid d - a$ being true (see example below).

Let us first note the following statement that will be frequently used in the sequel:

Fact: $\{J_a(d), J_{d-a}(d)\}$ is a bipartition of $\{1, \dots, d - 1\}$. (Easy.)

The following lemma looks like a simple and quite basic statement :

Lemma 3.2 *Let $d, a, b \in \mathbb{N}$, $0 < a, b < d/2$, $\gcd(a, d) = \gcd(b, d) = 1$. If $J_a(d) \subseteq J_b(d)$, then $a \mid b$.*

Proof. Let $a, b \in \mathbb{N}$, $0 < a, b < d/2$, $J_a(d) \subseteq J_b(d)$. Then $a \leq b$. Suppose a does not divide b , and let us show $J_a \setminus J_b \neq \emptyset$. We have then $2 \leq a < b$, and $J_a \setminus J_b \neq \emptyset$ means exactly the existence of $k \in \{1, \dots, a - 1\}$ such that $\lfloor kd/a \rfloor \notin J_b$ (see (*)).

Claim : Let $k \in \{1, \dots, a - 1\}$. Then $\lfloor kd/a \rfloor \notin J_b$ if and only if both

$$\frac{\text{mod}(kd, a)}{\text{mod}(kb, a)} < \frac{d}{b}, \text{ and } \frac{a - \text{mod}(kd, a)}{a - \text{mod}(kb, a)} < \frac{d}{b} \text{ hold.}$$

This statement looks somewhat scaring, but we will see that it expresses exactly what $\lfloor kd/a \rfloor \notin J_b$ means if one exploits (*) for b , and (**) for a :

Indeed, then $\lfloor kd/a \rfloor \notin J_b$ if and only if for all $i = 1, \dots, b - 1$: $id/b \notin (\lfloor kd/a \rfloor, \lfloor kd/a \rfloor + 1]$. Let us realize, that instead of checking this condition for all i , it is enough to check it for those possibilities for i for which id/b has a chance of being in $(\lfloor kd/a \rfloor, \lfloor kd/a \rfloor + 1]$:

Since $\frac{kd}{a} = \frac{kb}{a} \frac{d}{b}$ and hence $\lfloor \frac{kb}{a} \rfloor \frac{d}{b} \leq \frac{kd}{a} \leq \lceil \frac{kb}{a} \rceil \frac{d}{b}$, there are only two such possibilities for i : $\lfloor \frac{kb}{a} \rfloor$ and $\lceil \frac{kb}{a} \rceil$. In other words, $\lfloor \frac{kd}{a} \rfloor \notin J_b$ if and only if

$$\lfloor \frac{kb}{a} \rfloor \frac{d}{b} < \lfloor \frac{kd}{a} \rfloor, \text{ or } \lceil \frac{kb}{a} \rceil \frac{d}{b} > 1 + \lfloor \frac{kd}{a} \rfloor.$$

Subtract from these inequalities the equality $\frac{kb}{a} \frac{d}{b} = \frac{kd}{a}$, and apply $\lfloor p/q \rfloor = \frac{p - \text{mod}(p, q)}{q}$:

$$-\frac{\text{mod}(kb, a)}{a} \frac{d}{b} < -\frac{\text{mod}(kd, a)}{a},$$

which is the first inequality of the claim, and

$$\frac{a - \text{mod}(kb, a)}{a} \frac{d}{b} > 1 - \frac{\text{mod}(kd, a)}{a},$$

which is the second inequality. The claim is proved.

Let $g := \gcd(a, b)$. The values $\text{mod}(ib, a)$ ($i \in \{0, 1, \dots, a-1\}$) are from the set $\{jg : j = 0, \dots, (a/g) - 1\}$, and each number in this set is taken g times. Depending on whether a/g is even or odd, $a/2$ or $a - g/2$ is in this set.

So there exist g different values of i for which $a/2 - g/2 \leq \text{mod}(ib, a) \leq a/2$ and since for all of these g values $\text{mod}(id, a)$ is different (because of $\gcd(a, d) = 1$), for at least one of them $\text{mod}(id, a) \leq a - g$. For this i we have $\frac{\text{mod}(id, a)}{\text{mod}(ib, a)} \leq \frac{a-g}{a/2-g/2} = 2$, and since $a - \text{mod}(ib, a) \geq a/2$, $\frac{a-\text{mod}(ib, a)}{a-\text{mod}(id, a)} \leq 2$ holds as well. Since $d/b > 2$ by assumption, the condition of the claim is satisfied and we conclude $[id/a] \notin J_b$. $\square$

Corollary 3.3 *Let $d, a, b \in \mathbb{N}$, $d/2 < a, b < d$, $\gcd(a, d) = \gcd(b, d) = 1$. If $J_a(d) \subseteq J_b(d)$, then $d - b \mid d - a$.*

Indeed, according to the Fact proved before the Lemma, $J_a(d) \subseteq J_b(d)$ implies $J_{d-b}(d) \subseteq J_{d-a}(d)$, and clearly, $0 < d - b, d - a < d/2$, $\gcd(d - b, d) = \gcd(d - a, d) = 1$. Hence the Lemma can be applied to $d - b, d - a$ and it establishes $d - b \mid d - a$.

The following example shows that the Lemma or its corollary are not necessarily true if $a < d/2$, $b > d/2$, even if the condition of the lemma is ‘asymptotically true’ ($\lim d/b = 2$ if $k \rightarrow \infty$): let $k \in \mathbb{N}$, $k \geq 3$; $d := 6k - 1$, $a = 3$, $b = 3k + 4$.

Then $J_a = \{2k - 1, 4k - 1\} \subseteq J_b$ – let us check for instance $2k - 1 \in J_b$: $(2k - 1)b = 6k^2 + 5k - 4 = (6k - 1)k + 6k - 4 \equiv 6k - 4 \pmod{6k - 1}$. Since $3k + 4 > (6k - 1) - (6k - 4) = 3$, $2k - 1$ is a jump coefficient of $b = 3k + 4 \pmod{d}$.

For $k = 2$ we do not get a real example: $a = 3$, $b = 10$, $d = 11$; $J_a \subseteq J_b$ is true, and 3 is not a divisor of 10, but the corollary applies to $d - a = 8$ and $d - b = 1$. One gets the smallest example with $J_a \subseteq J_b$ and neither $a \mid b$ nor $d - b \mid d - a$ by substituting $k = 3$: then $a = 3$, $b = 13$, $d = 17$.

4 A Polynomial Certificate

In the introduction we formulated a somewhat more general question than the emptiness of lattice simplices: *given a linearly independent set $V := \{v_1, \dots, v_n\}$ of integer vectors, is there a $\text{par}(V)$ -coefficient vector whose (weighted) sum is smaller than a pre-given value.* If all the coefficients are 1, and the pre-given value is $d := \det(V)$ we get back the problem of the nonemptiness of $\text{conv}(0, v_1, \dots, v_n)$.

For integer weights $0 \leq a_1, \dots, a_n \leq d - 1$, the congruence $\sum_{i=1}^n a_i \lambda_i \equiv 0 \pmod{d}$ ($\lambda_1, \dots, \lambda_n \in \mathbb{N}$) implies $\sum_{i=1}^n a_i \lambda_i \geq d$, unless $a_i \lambda_i = 0$ for all $i = 1, \dots, n$. If the congruence holds for all $(\lambda_1, \dots, \lambda_n) \in \text{par}(V)$, then the inequality

also holds, except if $\lambda_i = 0$ for all $i = 1, \dots, n$ for which $a_i > 0$. We suppose that this exception does not occur. (This is automatically the case if all proper faces of $\text{par}(V)$ are empty, or if $a_i > 0$ for all $i = 1, \dots, n$.) Then, in order to certify the validity of the above inequality for the entire $\text{par}(V)$, one only has to check the congruence to hold for a generating set.

Such inequalities (induced exactly by the ‘orthogonal’ space to $G(V) \bmod d$) can then be combined in the usual way of linear (or integer) programming, in order to yield new inequalities. We do not have an example where this procedure would not provide a ‘short’ certificate for a lattice simplex to be empty, or more generally, for the inequality $\sum_{i=1}^n \lambda_i/d > k$ ($k \in \mathbb{N}$) to hold.

By the symmetry of the parallelepiped, the maximum of k for which such an inequality can be satisfied is $k = n/2$.

For this extreme case (which occurs in the open cases of the integer Caratheodory property – and slightly changes for odd n) we conjecture that the simplest possible ‘good certificate’ can work:

Conjecture 4.1 *Let $e \in \{0, 1\}$, $e \equiv n \bmod 2$, and $V = \{v_1, \dots, v_n\} \subseteq \mathbb{Z}^n$ be linearly independent. Then $\sum_{i=1}^n \lambda_i \geq \lfloor n/2 \rfloor d + 1 - e$ holds for every $\lambda = (\lambda_1, \dots, \lambda_n) \in G(V)$, if and only if there exists $\alpha = (\alpha_1, \dots, \alpha_n) \in G(V)$, $\gcd(\alpha_i, d) = 1$ ($i = 1, \dots, n$), and a set $\mathcal{P}$ of $\lfloor n/2 \rfloor$ disjoint pairs in $\{1, \dots, n\}$ such that for each pair $\{p, q\} \in \mathcal{P}$: $\alpha_p + \alpha_q = d$.*

The sufficiency of the condition is easy: α is a generator; for all $\lambda \in G(V)$, $\lambda > 0$; then $0 < \lambda_p + \lambda_q < 2d$ for every $\{p, q\} \in \mathcal{P}$, and $d \mid \lambda_p + \lambda_q$, so $\lambda_p + \lambda_q = d$ for every $\lambda \in G(V)$; $\sum_{i=1}^n \lambda_i \geq |\mathcal{P}| + 1 - e = \lfloor n/2 \rfloor d + 1 - e$ follows.

Conversely, if $\sum_{i=1}^n \lambda_i \geq \lfloor n/2 \rfloor d + 1 - e$ for every $\lambda \in G(V)$, then $\text{par}(V \setminus v_i) = \{0\}$ ($i = 1, \dots, n$) is obvious: if say $\text{par}(V \setminus v_n) \neq \{0\}$, then by symmetry, there exists $\lambda \in \text{par}(V \setminus e_n) \neq \{0\}$, $\sum_{i=1}^n \lambda_i/d \leq (n-1)/2$, a contradiction. So $G(V)$ is cyclic. In order to prove the conjecture, we have to prove that $\sum_{i=1}^n \lambda_i \geq \lfloor n/2 \rfloor d + 1 - e$ for every $\lambda = (\lambda_1, \dots, \lambda_n) \in G(V)$ implies $\lambda_p + \lambda_q = d$ for all $\text{par}(V)$ coefficient, or equivalently, for a generator.

We show this Conjecture in the special case $n \leq 4$, when it is equivalent to a celebrated result of White [15]. Instead of using White’s theorem, we provide a new, simpler proof based on Lemma 3.2. We hope that these results will be also useful in more general situations.

An integer simplex $S \subseteq \mathbb{R}^2$ is empty if and only if its two nonzero vertices form an empty parallelepiped, that is, if and only if they define a unimodular matrix. (This is trivial from the Hermite normal form, or just by the symmetry of parallelepipeds.)

Let now $n = 3$, and let $A, B, C \in \mathbb{Z}^3$ be the nonzero vertices of the simplex $S \subseteq \mathbb{R}^3$. It follows from the result on $n \leq 2$ applied to the facets of S after applying Lemma 3.1, that $G(V)$ is cyclic, and then the input of the problem can be given in a shorter and more symmetric form: we suppose that the input consists of only three numbers, the $V := \{A, B, C\}$ -coordinates of a generator (α, β, γ) of $G(V)$ (instead of the vectors A, B, C themselves). White’s theorem (see [15] or [10]) can be stated as follows:

Theorem 4.2 *Let $S \subseteq \mathbb{R}^3$ be an integer simplex with vertices $O = 0 \in \mathbb{R}^3$, and A, B, C linearly independent vectors, $d := \det(A, B, C)$, $V := \{A, B, C\}$. The following statements are equivalent:*

- (i) $\text{par}(A, B) = \text{par}(B, C) = \text{par}(A, C) = \{0\}$, and there exists a $\text{par}(V)$ -coefficient vector (α, β, γ) such that $\gcd(\alpha, d) = \gcd(\beta, d) = \gcd(\gamma, d) = 1$, moreover the sum of two of α, β, γ is equal to d , and the third is equal to 1.
- (ii) $\text{par}(A, B) = \text{par}(B, C) = \text{par}(A, C) = \{0\}$, and for every $\text{par}(V)$ -coefficient vector (α, β, γ) , after possible permutation of the coordinates $\text{mod}(\alpha, d) + \text{mod}(\beta, d) = d$ ($i = 1, \dots, d-1$).
- (iii) S is empty.

Proof. If (i) holds, then $(\alpha, \beta, \gamma) \in G(V)$ is a generator, so (ii) also holds. If (ii) holds, then for all $x \in \text{par}(A, B, C)$ the sum of the first two V -coefficients is divisible by d , and it follows that it is equal to d ; since $\text{par}(A, B) = \{0\}$, the third V -coordinate of x is nonzero, so the sum of the V -coefficients is greater than d which means exactly that $x \notin S$. So S is empty.

The main part of the proof is (iii) implies (i). We follow the proof of [14] Theorem 2.2:

Let $S \subseteq \mathbb{R}^3$ be an empty integer simplex. Then every face of S is also integer and empty. Therefore (since the faces are two-dimensional) $\text{par}(V') = \{0\}$ for every proper subset $V' \subset V$. Now by Lemma 3.1, the group $G(V)$ is cyclic. Let the $\text{par}(V)$ -coefficient (α, β, γ) be a generator.

Claim 1: $d < \text{mod}(i\alpha, d) + \text{mod}(i\beta, d) + \text{mod}(i\gamma, d) < 2d$ ($i = 1, \dots, d-1$).

Indeed, since S is empty, $\text{mod}(i\alpha, d) + \text{mod}(i\beta, d) + \text{mod}(i\gamma, d) > d$, and $(d - \text{mod}(i\alpha, d)) + d - \text{mod}(i\beta, d) + d - \text{mod}(i\gamma, d) > d$, for all $i = 1, \dots, d-1$.

Claim 2: There exists a generator $(\alpha, \beta, \gamma) \in G(V)$ such that $\alpha + \beta + \gamma = d + 1$.

Note that $\text{mod}(i\alpha, d) + \text{mod}(i\beta, d) + \text{mod}(i\gamma, d)$ is $\text{mod } d$ different for different $i \in \{1, \dots, d-1\}$, because if for j, k , $0 \leq j < k \leq d-1$ the values are equal, then for $i = k - j$ the expression would be divisible by d , contradicting Claim 1. So $\{\text{mod}(i\alpha, d) + \text{mod}(i\beta, d) + \text{mod}(i\gamma, d) : i = 1, \dots, d-1\}$ is the same as the set $\{d+1, \dots, 2d-1\}$, in particular there exists $k \in \{1, \dots, d-1\}$ such that $\text{mod}(k\alpha, d) + \text{mod}(k\beta, d) + \text{mod}(k\gamma, d) = d + 1$. Then clearly, $\text{mod}(ik\alpha, d) + \text{mod}(ik\beta, d) + \text{mod}(ik\gamma, d) = d + i$, so $(k\alpha, k\beta, k\gamma)$ is also a generator, and the claim is proved.

Choose now (α, β, γ) be like in Claim 2.

Claim 3. Each $i = 1, \dots, d-1$ is jump-coefficient of exactly one of α, β and γ .

Indeed, because of Claim 1 we have $\text{mod}((i+1)\alpha, d) + \text{mod}((i+1)\beta, d) + \text{mod}((i+1)\gamma, d) = \text{mod}(i\alpha, d) + \text{mod}(i\beta, d) + \text{mod}(i\gamma, d) + \alpha + \beta + \gamma - d$ and the claim follows.

Fix now the notation so that $\alpha \geq \beta \geq \gamma$. If we apply Claim 3 to $i = 1$ we get that $\alpha > d/2$, $\beta < d/2$, $\gamma < d/2$.

Hence Lemma 3.2 can be applied to $d - \alpha, \beta, \gamma$: Claim 3 means $J_\beta \cap J_\alpha = \emptyset$, $J_\gamma \cap J_\alpha = \emptyset$, whence, because of the Fact noticed before Lemma 3.2 $J_\beta, J_\gamma \subseteq$

$J_{d-\alpha}$. So by Lemma 3.2 $\beta, \gamma \mid d - \alpha$. If both β and γ are proper divisors, then they are both smaller than the half or $d - \alpha$, that is, $\beta + \gamma \leq d - \alpha$, contradicting $\alpha + \beta + \gamma = d + 1$.

So $\beta = d - \alpha$, and then $\gamma = 1$ finishing the proof of the theorem. $\square$

Note that the above proof contains most of the proof of [14] Theorem 2.2 and more: the key-statement of that proof is the existence of a $\text{par}(V)$ -coefficient (α, β, γ) such that $\alpha + \beta + \gamma = d + 1$. The above proof sharpens that statement by adding that $\gamma = 1$ in this $\text{par}(V)$ -coefficient. This additional fact implies some simplifications for proving the main result of [14]. Here we omit these, we prefer to deduce the following, in fact equivalent versions of Theorem 4.2 ([11],[15],[12]). It turns out that the various versions are the same modulo row permutations in the Hermite normal form of a 3×3 matrix that has two different types of rows:

Corollary 4.3 $S = \text{conv}(0, A, B, C) \subseteq \mathbb{R}^3$, $(A, B, C \in \mathbb{Z}^3)$ is empty if and only if the Hermite normal form of one of the (six) 3×3 matrices M whose rows are A, B, C in some order has rows $(1, 0, 0)$, $(0, 1, 0)$, $(a, d - a, d)$, $(\gcd(a, d) = 1)$.

Proof. The if part is obvious, since for an arbitrary $\text{par}(V)$ -coefficient vector (α, β, γ) : $\alpha + \beta = d$ and $\gamma > 0$, so $\alpha/d + \beta/d + \gamma/d > 1$.

The only if part is an obvious consequence of the theorem: let the two particular rows mentioned in Theorem 4.2 (i) be the first two rows of a matrix M , and let the remaining row be the third. Then M is a 3×3 matrix, and with the notation of Theorem 4.2, the rows of the Hermite normal form of M are the following : the first two rows are $(1, 0, 0)$, $(0, 1, 0)$ since the parallelepiped generated by any two nonzero extreme rays is $\{0\}$; the third is $(d - \alpha, d - \beta, d)$. Since by Theorem 4.2 $\alpha + \beta = d$, the statement follows. $\square$

Corollary 4.4 $S = \text{conv}(0, A, B, C) \subseteq \mathbb{R}^3$ $(A, B, C \in \mathbb{Z}^3)$ is empty if and only if the Hermite normal form of one of the (six) 3×3 matrices M whose rows are A, B, C in some order has rows $(1, 0, 0)$, $(0, 1, 0)$ and $(1, b, d)$, $(\gcd(b, d) = 1)$.

Proof. Again, the if part is obvious, because in an arbitrary $\text{par}(V)$ -coefficient vector the sum of the first and third coordinate is d , and the second coordinate is positive.

The proof of the only if part is also the same as that of the previous corollary, with the only difference that now we let the two particular rows mentioned in Theorem 4.2 (i) be the first and third row of M . $\square$

Corollary 4.5 The integer simplex $S \subseteq \mathbb{R}^n$, $n \leq 3$ is empty, if and only if $W(P) = 1$ for all faces P of S (that is, for $P = S$ and for all P which is any of the facets or edges of S).

Proof. If $n \leq 2$ the statement is obvious (see above). The only if part follows from the previous corollary: after applying a unimodular transformation, the three nonzero vertices of S will be $(1, 0, 0)$, $(0, 1, 0)$ and $(1, b, d)$. The vector $w := (1, 0, 0)$ shows that the width of S is 1.

Conversely, suppose $W(P) = 1$ for every face P of the integer simplex S . Suppose indirectly $v \in S \cap \mathbb{Z}^n$ is not a vertex of S . Because of the statement for $n \leq 2$, v does not lie on a face $P \subset S$, $P \neq S$. For a vector $w \in \mathbb{Z}^n$ defining the width, $\min\{w^T x : x \in S\} < w^T v < \max\{w^T x : x \in S\}$, because the vectors achieving the minimum or the maximum constitute a face of S . But then $W(S) = \max\{w^T x : x \in S\} - \min\{w^T x : x \in S\} = \max\{w^T x : x \in S\} - w^T v + w^T v - \min\{w^T x : x \in S\} \geq 1 + 1 = 2$, a contradiction. $\square$

Corollary 4.6 *Conjecture 4.1 is true for $n \leq 4$.*

Proof. For $n \leq 3$ the statement is a consequence of Theorem 4.2.

Let $n = 4$. As noticed after Conjecture 4.1, we only have to prove that $\sum_{i=1}^n \lambda_i \geq 2d$ for every $\lambda \in G(V)$ implies that for some generator α (possibly after permuting the coordinates) $\alpha_1 + \alpha_2 = d$, $\alpha_3 + \alpha_4 = d$.

Let $\alpha \in G(V)$ be a generator, and suppose without loss of generality $\alpha_4 = d - 1$. (This holds after multiplying α by a number relatively prime to d .) Then Claim 2 and 3 of the proof of Theorem 4.2 hold with the choice $\alpha := \alpha_1$, $\beta := \alpha_2$, $\gamma := \alpha_3$, and the proof can be finished in the same way. $\square$

This argument can be copied for proving that Conjecture 4.1 holds true for $n = 2k - 1$ if and only if it is true for $n = 2k$.

5 Computing the Width of Polytopes

In spite of some counterexamples (see Section 1), the width and the number of lattice points of a lattice simplex are correlated, and some of the remarks above are about this relation. It is interesting to note that the complexity of computing these two numbers seem to show some analogy: it is hard to compute the number of integer points of a polytope, but according to a recent result of Barvinok[2] this problem is polynomially solvable if the dimension is bounded; we show below that to compute the width of quite particular simplices is already NP-hard, however, there is a simple algorithm that finds the width in polynomial time if the dimension is fixed. The proofs are quite easy:

Theorem 5.1 *Let $a \in \mathbb{N}^n$. It is NP-complete to decide whether the width of $\text{conv}(e_1, \dots, e_n, a)$ is at most 1.*

Proof. The defined problem is clearly in NP. We reduce PARTITION to this problem. Let the input of a PARTITION problem be a vector $a \in \mathbb{N}^n$, where $-a_n := \sum_{i=1}^{n-1} a_i/2$ and we can suppose without loss of generality, $a_i > 3$ ($i = 1, \dots, n$). (We multiply by 4 an arbitrary instance of PARTITION. The question of PARTITION is whether there exists a subset $I \subseteq \{1, \dots, n-1\}$ such that $\sum_{i \in I} a_i = -a_n$, and this is unchanged under multiplication of all the data by a scalar.)

It is easy to see that the width of $P := \text{conv}(e_1, \dots, e_n, a)$ is at most 1 if and only if the defined PARTITION problem has a solution.

Indeed, if $I \subseteq \{1, \dots, n-1\}$ is a solution, then for the vector $w \in \mathbb{Z}^n$ defined by $w_i := 1$ if $i \in I \cup \{n\}$, and $w_i := 0$ if $i \notin I \cup \{n\}$ we have $w^T e_i \leq 1$ ($i = 1, \dots, n$), $w^T a = 0$.

Conversely, let $W(P) \leq 1$, and let $w \in \mathbb{N}^n$ be the vector defining the width. Then $|w^T x - w^T y| \leq 1$, $x, y \in \{e_1, \dots, e_n\}$ means that the value of w_i ($i = 1, \dots, n$) is one of two consecutive integers, and these are nonnegative without loss of generality: for some $k \in \mathbb{N}$ and $K \subseteq \{1, \dots, n\}$, $w_i := k + 1$ if $i \in K$ and $w_i := k$ if $i \notin K$; but then $w^T a = k \sum_{i=1}^n a_i + \sum_{i \in K} a_i$. Since w defines the width, $|w^T a| \in \{k, k + 1\}$, and an easy computation shows that this is possible only if $k = 0$ and $K \setminus \{n\}$ is a solution of the partition problem. $\square$

Let us consider a polytope $P := \{x \in \mathbb{R}^n : Ax \leq b\}$ where A is a matrix with n columns. Denote the i -th row of A by a_i , and the corresponding right hand side by b_i . For simplicity (and without loss of generality) suppose that P is full dimensional. If v is a vertex of P , denote $C_v := \text{cone}(a_i : a_i^T v = b_i)$ and let $X_{u,v}$ denote the set of extreme rays of the cone $C_u \cap -C_v$ so that for every $x \in X_{u,v} : x = (x_1, \dots, x_n) \in \mathbb{Z}^n$, $x_i \in \mathbb{Z}$, ($i = 1, \dots, n$), and $\gcd(x_1, \dots, x_n) = 1$. Let

$$Z_{u,v} := \bigcup \{\text{conv}(X \cup \{0\}) : X \subseteq X_{u,v}, X \text{ is a basis of } \mathbb{R}^n\} \cap \mathbb{Z}^n.$$

Theorem 5.2 $W(P) = \min w^T(u - v)$ over all pairs of vertices u, v of S , and all $w \in Z_{u,v}$.

Proof. By definition, the width of P is the minimum of $\max\{c^T(u - v) : u, v \in P\}$ over all vectors $c \in \mathbb{Z}^n \setminus \{0\}$. Let w be the minimizing vector. It follows from the theory of linear programming, that u, v can be chosen to be vertices of P which maximize and respectively minimize the linear objective function $w^T x$; moreover, by the duality theorem of linear programming, w is then a nonnegative combination of vectors in C_u and also a nonnegative combination of vectors in $-C_v$; that is, $w \in C_u \cap -C_v$. Fix now u and v to be these two vertices of P . All that remains to be proved is:

Claim. For some linearly independent subset $X \subseteq X_{u,v}$, $w \in \text{conv}(X \cup \{0\}) \cap \mathbb{Z}^n$.

Indeed, since $w \in C_u \cap -C_v$, we know by Caratheodory's theorem that $w = \sum_{x \in X} \lambda_x x$ for some linearly independent $X \subseteq X_{u,v}$ and $\lambda_x \in \mathbb{R}$, ($x \in X$). In order to prove the claim, we have to prove that $\sum_{x \in X} \lambda_x \leq 1$. If not: $w^T(u - v) = (\sum_{x \in X} \lambda_x x)^T(u - v) \geq (\sum_{x \in X} \lambda_x)(\min_{x \in X} x^T(u - v)) > \min_{x \in X} x^T(u - v)$.

Let this minimum be achieved in $x_0 \in X$, so we proved $w^T(u - v) > x_0^T(u - v)$. Since $x_0 \in C_u \cap -C_v$, $\max\{x_0^T(u' - v') : u', v' \in P\}$ is achieved for $u' = u$ and $v' = v$. But then $w^T(u - v) > x_0^T(u - v)$ contradicts the definition of w . $\square$

Theorem 5.2 provides a polynomial algorithm for computing the width if n is fixed: all the extreme rays and facets of $C_u \cap -C_v$ can be computed in polynomial time; since $|X_{u,v}|$ is bounded by a polynomial, it contains a polynomial number of subsets X of fixed size n ; for every X we solve a mixed integer program searching

for an integer point in $\text{conv}(X \cup \{0\})$ but not in $X \cup \{0\}$. Mixed integer programs can be solved in polynomial time by Lenstra [9], see also Schrijver [13], page 260.

Haase and Ziegler [5] present $W(S)$ where S is a lattice simplex as the optimal value of a direct integer linear program. Their method is much simpler, and probably quicker. The finite set of points ('finite basis') provided by Theorem 5.2 can be useful for presenting a finite list of vectors that include a width-defining $w \in \mathbb{Z}^n$, for arbitrary polytopes.

The negative results of the paper do not exclude that the emptiness of integer simplices, and the width of empty integer simplices are decidable in polynomial time. The positive results show some relations between these notions, involving both complexity and bounds.

Acknowledgment: I am thankful to Jean-Michel Kantor for introducing me to the notions and the references of the subject; to Imre Bárány and Bernd Sturmfels for further helpful discussions.

References

1. W. Banaszczyk, A.E. Litvak, A. Pajor, S.J. Szarek, The flatness theorem for non-symmetric convex bodies via the local theory of Banach spaces, Preprint 1998.
2. A. Barvinok, A polynomial time algorithm for counting integral point in polyhedra when the dimension is fixed, *Math. Oper. Res.*, 19 (1994), 769–779.
3. J.W.S. Cassels, *An Introduction to the Geometry of Numbers*, Springer, Berlin, 1959.
4. W. Bienia, L. Goddyn, P. Gvozďjak, A. Sebő, M. Tarsi, Flows, View Obstructions, and the Lonely Runner, *J. Comb. Theory/B*, Vol 72, No 1, 1998.
5. C. Haase, G. Ziegler, On the maximal width of empty lattice simplices, preprint, July 1998/January 1999, 10 pages, *European Journal of Combinatorics*, to appear.
6. R. Kannan, L. Lovász, Covering minima and lattice-point-free convex bodies, *Annals of Mathematics*, 128 (1988), 577–602.
7. L. Lovász and M. D. Plummer, *Matching Theory*, Akadémiai Kiadó, Budapest, 1986.
8. J-M. Kantor, On the width of lattice-free simplices, *Composition Mathematica*, 1999.
9. H.W. Lenstra Jr., Integer programming with a fixed number of variables, *Mathematics of Operations Research*, 8, (1983), 538–548.
10. B. Reznick, Lattice point simplices, *Discrete Mathematics*, 60, 1986, 219–242.
11. J.E. Reeve, On the volume of lattice polyhedra, *Proc. London Math. Soc.* (3) 7 (1957), 378–395.
12. H. Scarf, Integral polyhedra in three space, *Math. Oper. Res.*, 10 (1985), 403–438.
13. A. Schrijver, 'Theory of Integer and Linear Programming', Wiley, Chichester, 1986.
14. A. Sebő, Hilbert bases, Caratheodory's theorem and Combinatorial Optimization, IPCO1, (R. Kannan and W. Pulleyblank eds), University of Waterloo Press, Waterloo 1990, 431–456.
15. G. K. White, Lattice tetrahedra, *Canadian J. Math.* 16 (1964), 389–396.

On Optimal Ear-Decompositions of Graphs

Zoltán Szigeti

Equipe Combinatoire, Université Paris 6, 75252 Paris, Cedex 05, France
Zoltan.Szigeti@ecp6.jussieu.fr

Abstract. This paper can be considered as a continuation of a paper [7] of the author. We consider optimal ear-decompositions of graphs that contain even ears as few as possible. The ear matroid of a graph was introduced in [7] via optimal ear-decompositions. Here we give a simple description of the blocks of the ear matroid of a graph. The second goal of this paper is to point out how the structural result in [7] implies easily the Tight Cut Lemma of Edmonds, Lovász and Pulleyblank. Moreover, we propose the investigation of a new class of graphs that generalizes matching-covered graphs. A graph is called φ -covered if each edge may lie on an even ear of an optimal ear-decomposition. Several theorems on matching-covered graphs will be generalized for φ -covered graphs.

1 Introduction

This paper is concerned with matchings, matroids and ear-decompositions. The results presented here are closely related to the ones in [7]. As in [7], we focus our attention on ear-decompositions (called optimal) that have minimum number $\varphi(G)$ of even ears. A. Frank showed in [3] how an optimal ear-decomposition can be constructed in polynomial time for any 2-edge-connected graph. For an application we refer the reader to [1], where we gave a 17/12-approximation algorithm for the minimum 2-edge-connected spanning subgraph problem. $\varphi(G)$ can also be interpreted as the minimum size of an edge-set (called critical making) whose contraction leaves a factor-critical graph, because, by a result of Lovász (see Theorem 1), $\varphi(G) = 0$ if and only if G is factor-critical. It was shown in [7] (see also [8]) that the minimal critical making edge sets form the bases of a matroid. We call the matroid as the ear-matroid of the graph and it will be denoted by $\mathcal{M}_G$. For a better understanding of this matroid we give here a simple description of the blocks of $\mathcal{M}_G$. As we shall see, two edges belong to the same block of the ear-matroid if and only if there exists an optimal ear-decomposition so that these two edges are contained in the same even ear. Let us denote by $D(G)$ the graph obtained from G by deleting each edge which is always in an odd ear in all optimal ear-decompositions of G . We shall show that the edge sets of the blocks of this graph $D(G)$ coincide with the blocks of $\mathcal{M}_G$.

Let us turn our attention to matching theory. Matching-covered graphs and saturated graphs have $\varphi(G) = 1$. By generalizing the Cathedral Theorem of Lovász [6] on saturated graphs we proved in [7] a structure theorem on graphs with $\varphi(G) = 1$. As a new application of that result we present here a simple

proof for the Tight Cut Lemma due to Edmonds, Lovász and Pulleyblank [2]. Edmonds et al. determined in [2] the dimension of the perfect matching polytope of an arbitrary graph. They used the so-called brick decomposition and the most complicated part was to show that the result is true for bricks. In this case the problem is equivalent to saying that no brick contains a non-trivial tight cut. This is the Tight Cut Lemma. Their proof for this lemma contains some linear programming arguments, the description of the perfect matching polytope, the uncrossing technique and some graph theoretic arguments. Here we provide a purely graph theoretic proof.

Let us call a graph φ -covered if for each edge e there exists an optimal ear-decomposition so that e is contained in an even ear. We shall generalize matching-covered graphs via the following fact.

($\times$) A graph G is matching-covered if and only if G/e is factor-critical for each edge e of G , in other words, G is φ -covered and $\varphi(G) = 1$.

Thus φ -covered graphs can be considered as the generalization of matching-covered graphs. We shall extend several results on matching-covered graphs for φ -covered graphs. For example Little's theorem combining by Lovász and Plummer's result states that if e and f are two arbitrary edges of a matching-covered graph G , then G has an optimal ear-decomposition such that the first ear is even and it contains e and f . This is also true for φ -covered graphs. Lovász and Plummer [6] proved that each matching-covered graph has a 2-graded ear-decomposition. For a simple proof for this theorem see [9]. This result may also be generalized for φ -covered graphs.

2 Definitions and Preliminaries

Let $G = (V, E)$ be a (2-edge-connected) graph. An *ear-decomposition* of G is a sequence $G_0, G_1, \dots, G_n = G$ of subgraphs of G where G_0 is a vertex and each G_i arises from G_{i-1} by adding a path P_i for which the two end-vertices (they are not necessarily distinct) belong to G_{i-1} while the inner vertices of P_i do not. This means that the graph G can be written in the following form: $G = P_1 + P_2 + \dots + P_n$, where the paths P_i are called the *ears* of this decomposition. Note that we allow closed ears, for example the first ear (*starting ear*) is always a cycle. An ear is *odd* (resp. *even*) if its length is odd (resp. even). If each ear is odd then we say that it is an *odd ear-decomposition*. For 2-edge-connected graphs $\varphi(G)$ is defined to be the minimum number of even ears in an ear-decomposition of G . An ear-decomposition is said to be *optimal* if it has exactly $\varphi(G)$ even ears.

A graph G is called *factor-critical* if for each vertex $v \in V(G)$, $G - v$ has a perfect matching.

Theorem 1. Lovász [5]

a.) A graph G is factor-critical if and only if $\varphi(G) = 0$.

b.) Let H be a connected subgraph of a graph G . If H and G/H are factor-critical, then G is factor-critical. $\square$

We say that an edge set of a graph G is *critical making* if its contraction leaves a factor-critical graph. An edge of a graph G is *allowed* if it lies in some perfect matching of G . If G has a perfect matching then $N(G)$ denotes the subgraph of G induced by the allowed edges of G . A connected graph G is *matching covered* if each edge of G is allowed. The following theorem is obtained by combining the result of Little [4] and a result of Lovász and Plummer [6].

Theorem 2. *Let e and f be two arbitrary edges of a matching-covered graph G . Then G has an ear-decomposition such that only the first ear is even and it contains e and f .* $\square$

We call a graph *almost critical* if $\varphi(G) = 1$. By Theorem 2, matching covered graphs are almost critical. We mention that if G is almost critical then each critical making edge of G is allowed. Let G be an arbitrary graph. If $X \subseteq V(G)$, then the subgraph of G induced by X is denoted by $G[X]$. The graph obtained from G by contracting an edge e of G will be denoted by G/e . By the subdivision of an edge e we mean the operation which subdivides e by a new vertex. The new graph will be denoted by $G \times e$. We say that an edge e of G is φ -*extreme* if $\varphi(G/e) = \varphi(G) - 1$, and more generally, an edge set F of G is φ -*extreme* if $\varphi(G/F) = \varphi(G) - |F|$. We proved in Lemma 1.1 in [8] that for a 2-edge-connected graph G ,

$$\varphi(G \times F) = \varphi(G/F) \text{ if } F \text{ is a forest in } G. \quad (1)$$

A graph G is called φ -*covered* if each edge of G is φ -extreme. For a graph G , we denote by $D(G)$ the graph on $V(G)$ whose edges are exactly the φ -extreme edges of G . The ear matroid $\mathcal{M}(G)$ of a graph G was introduced in [8]. Its independent sets are the φ -extreme edge sets of G and its bases are exactly the minimum critical making edge sets. The set of bases of $\mathcal{M}(G)$ will be denoted by $\mathcal{B}(G)$. The *blocks* of a graph G are defined to be the maximal 2-vertex-connected subgraphs of G .

A connected component H of a graph G is called *odd* (*even*) if $|V(H)|$ is odd (even). A subgraph H of G is called *nice* if $G - V(H)$ has a perfect matching. Let H be a graph with a perfect matching. By Tutte's Theorem $c_o(H - X) \leq |X|$ for all $X \subseteq V(H)$. A vertex set X is called *barrier* if $c_o(H - X) = |X|$, where for $X \subseteq V(H)$, $c_o(H - X)$ denotes the number of odd components in $H - X$. A non-empty barrier X of H is said to be a *strong barrier* if $H - X$ has no even components, each of the odd components is factor-critical and the bipartite graph H_X obtained from H by deleting the edges spanned by X and by contracting each factor-critical component to a single vertex is matching covered. Let $G = (V, E)$ be a graph and assume that the subgraph H of G induced by $U \subseteq V$ has a strong barrier X . Then H is said to be a *strong subgraph* of G with strong barrier X if X separates $U - X$ from $V - U$ or if $U = V$.

Theorem 3. *Frank [3]*

- a.) *A graph has a strong subgraph if and only if it is not factor-critical.*
- b.) *Let H be a strong subgraph of a graph G . Then $\varphi(H) = 1$ and $\varphi(G/H) = \varphi(G) - 1$.* $\square$

It is easy to see that (*) if H is a strong subgraph (with strong barrier X) of an almost critical graph G , then $G - (V(H) - X)$ contains no critical making edge of G . The following lemma generalizes (*).

Lemma 1. *Let H be a strong subgraph with strong barrier X of a graph G . Then an edge e of G is φ -extreme in G if and only if e is φ -extreme either in H or in G/H . That is, $E(D(G)) = E(D(H)) \cup E(D(G/H))$.*

Proof. To see the if part let e be a φ -extreme edge either in H or in G/H . By Theorem 3/b, there exists an edge f in H and an edge set F in G/H of size $\varphi(G) - 1$ so that H/f and $(G/H)/F$ are factor-critical and by the fact that the φ -extreme edge sets are the independent sets of the ear matroid we may suppose that $F' := F \cup f$ contains e . Then, by Theorem 1/b, G/F' is factor-critical and since $|F'| = \varphi(G)$ it follows that $F' \in \mathcal{B}(G)$ that is e is φ -extreme in G .

The other direction will be derived from the following simple fact. Let e be a φ -extreme edge in G .

Proposition 1. *There exists $B_e \in \mathcal{B}(G)$ with $e \in B_e$ and $|B_e \cap E(G[V(H)])| = 1$.*

Proof. If at least one of the two end vertices of e is contained in one of the odd components of $H - X$, then let us denote this component by K , otherwise let K be an arbitrary odd component of $H - X$. Let f be a φ -extreme edge in H which connects K to X , such an edge exists by Theorem 4.3 in [3]. The proposition is true for f (see above the other direction), thus there exists a base $B_f \in \mathcal{B}(G)$ so that $f \in B_f$ and $|B_f \cap E(G[V(H)])| = 1$. e is φ -extreme edge in G that is it is independent in $\mathcal{M}(G)$. Thus it can be extended to a base $B_e \in \mathcal{B}(G)$ using elements in B_f . We show that B_e has the required properties. Clearly, $e \in B_e$. $|B_e \cap E(G[V(H)])| \leq 2$ by construction. Let us denote by X' (by H') the vertex set (subgraph) corresponding to X (H) in G/B_e . G/B_e is factor-critical because $B_e \in \mathcal{B}(G)$, whence trivially follows that $c_o(H' - X') < |X'|$. Thus, by construction, $|X| - 1 = c_o(H - X) - 1 \leq c_o(H' - X') \leq |X'| - 1 \leq |X| - 1$. Thus $c_o(H' - X') = c_o(H - X) - 1$ and $|X'| = |X|$. It follows that $|B_e \cap E(G[V(H)])| = 1$. $\square$

Let $D_e = B_e - E(G[V(H)])$. Let $G' := G/D_e$. Then, by the previous proposition, $\varphi(G') = 1$. We claim that H is a strong subgraph in G' . Otherwise, if X changes that is $|X|$ decreases, then the corresponding set X' violates the Tutte's condition in G' , a contradiction because, as it was mentioned in [3], any graph Q with $\varphi(Q) = 1$ has a perfect matching.

First suppose that $e \in E(G[V(H)])$. Then, by (*) and Lemma 2.5 in [7], the lemma is true. Now suppose that $e \in E(G/H)$. By Theorem 3/b, G'/H is factor-critical. Since $(G/H)/D_e = G'/H$ and $|D_e| = \varphi(G) - 1 = \varphi(G/H)$, $e \in D_e \in \mathcal{B}(G/H)$, that is e φ -extreme in G/H . $\square$

The first two statements of the following theorem were proved in [7]. Since the fourth statement is trivial, we prove here only the third one.

Theorem 4. *Let G be an almost critical graph. Then*

- (4.1) *$E(D(G))$ coincides with the edge set of one of the connected components of $N(G)$. This component will be denoted by $B(G)$.*
- (4.2) *$B(G)$ is matching-covered and $G/B(G)$ is factor-critical.*
- (4.3) *$V(B(G)) = \bigcap \{V(H) : H \text{ is a strong subgraph in } G\}$.*
- (4.4) *Let $u, v \in V(B(G))$ and let $T \subset V(G)$ separating u and v . Then $\delta(T)$ contains an allowed edge of G .*

Proof. Let us denote by U the vertex set on the right hand side in Theorem (4.3). By (*), $V(B(G)) \subseteq U$. To show the other direction let $v \in U$. Let e be an allowed edge of G incident to v . Let $G' := G/e$ and let us denote by u the vertex of G' corresponding to the two end vertices of e . $G' - u$ has a perfect matching because e is an allowed edge of G . If G' is not factor-critical, then it is easy to see (using Lemma 1.8 in [8]) that G' contains a strong subgraph H so that H does not contain u . It follows that H is a strong subgraph of G not containing v , which is a contradiction since $v \in U$. Thus G' is factor-critical, that is e is a critical making edge of G , thus $v \in V(B(G))$, and we are done. $\square$

3 The Ear Matroid

The blocks of a matroid $\mathcal{N}(G)$ are defined by an equivalence relation. For two elements e and f of $\mathcal{N}(G)$, $e \sim f$ if there exists a circuit in the matroid containing them. Observe that there exists a circuit in $\mathcal{M}(G)$ containing both e and f if and only if there exists a base $B \in \mathcal{B}(G)$ containing e so that $B - e + f$ is a base again. This is an equivalence relation and the blocks of $\mathcal{N}(G)$ are the equivalence classes of $\sim$. There is a close relation between the circuits of $\mathcal{M}(G)$ and the even ears of G .

Lemma 2. *The following are equivalent for the ear matroid $\mathcal{M}(G)$ of a 2-edge-connected graph G . Let e and f be two edges of G .*

- a.) *there exists an optimal ear-decomposition of G so that the first ear is even and it contains e and f .*
- b.) *e and f are in the same block of $\mathcal{M}(G)$.*

Proof. a.) $\implies$ b.) If e and f together can be in an even ear P of an optimal ear-decomposition then choosing one edge from each even ear (from P let e be chosen) we obtain the desired base.

b.) $\implies$ a.) Let B be a base containing e so that $B - e + f$ is a base again. Let $G' := G \times (B - e)$. Then, by (1), $\varphi(G') = \varphi(B - e) = 1$ and $e, f \in E(D(G'))$. By Theorem (4.2), $B(G')$ is matching covered, so by Theorem 2, $B(G')$ satisfies Lemma 2/a. By Theorem (4.2), $G'/B(G')$ is factor-critical, so by Theorem 1/a, it has an odd ear-decomposition. It follows that G' satisfies Lemma 2/a. Obviously, the ear-decomposition obtained provides the desired ear-decomposition of G . $\square$

Theorem 5. *Let G be a 2-vertex-connected φ -covered graph. Then the ear matroid $\mathcal{M}(G)$ has one block.*

Proof. We prove the theorem by induction on $\varphi(G)$. If $\varphi(G) = 1$, G is matching-covered, and then by Theorem 2 and Lemma 2, the theorem is true. In the rest of the proof we suppose that $\varphi(G) \geq 2$. Let H be a strong subgraph of G with strong barrier X . By Lemma 1, H and G/H are φ -covered, and, by Theorem 3/b, $\varphi(H) = 1$ and $\varphi(G/H) = \varphi(G) - 1$. Let G_1 be an arbitrary block of G/H .

i.) Let e_1 and e_2 be two arbitrary edges of H . Let $B \in \mathcal{B}(G/H)$. Then $(G/H)/B$ is factor-critical. H/e_1 and H/e_2 are factor-critical because $\varphi(H) = 1$ and H is φ -covered. Let $B' := B + e_1$. Then, by Theorem 1/b, G/B' and $G/(B' - e_1 + e_2)$ are factor-critical, that is B' and $B' - e_1 + e_2$ are in $\mathcal{B}(G)$, hence e_1 and e_2 belong to the same block of $\mathcal{M}(G)$.

ii.) Let e_1 and e_2 be two arbitrary edges of G_1 . By induction, e_1 and e_2 belong to the same block of $\mathcal{M}(G_1)$, thus there exists a base $B \in \mathcal{B}(G_1)$ so that $e_1 \in B$ and $B - e_1 + e_2 \in \mathcal{B}(G_1)$. For each block G_i of G/H different from G_1 let $B_i \in \mathcal{B}(G_i)$. Furthermore, let $f \in E(H)$. Finally, let $D := B \cup (\cup B_i) + f$. Note that $|D| = \varphi(G)$. Then, by Theorem 1/b, G/D and $G/(D - e_1 + e_2)$ are factor-critical, that is D and $D - e_1 + e_2 \in \mathcal{B}(G)$. Hence e_1 and e_2 belong to the same block of $\mathcal{M}(G)$.

iii.) Let e_1 and f_1 be two edges of G_1 so that the corresponding two edges in G are incident to two different vertices u and v of X . By the 2-vertex-connectivity of G , such edges exist. Let e_2 and f_2 be two edges of H incident to u and v respectively. By i.) ii.) and Lemma 2, there exists an optimal ear-decomposition $P_1 + P_2 + \dots + P_k$ ($P'_1 + P'_2 + \dots + P'_l$) of H (of G_1) so that e_2 and f_2 (e_1 and f_1) belong to the starting even ear. Furthermore, let $P''_1 + P''_2 + \dots + P''_m$ be an optimal ear-decomposition of $G/H/G_1$ so that the first ear contains the vertex corresponding to the contracted vertex set. Using these ear-decompositions we provide an optimal ear-decomposition of G so that the starting even ear will contain e_1 and e_2 . u and v divide P_1 (which is an even ear) into two paths D_1 and D_2 . Suppose D_1 contains e_2 . It is easy to see that

Proposition 2. D_1 and D_2 are of even length. □

Consider the following ear-decomposition of G : $(D_1 + P'_1) + D_2 + P_2 + \dots + P_k + P'_2 + \dots + P'_l + P''_1 + P''_2 + \dots + P''_m$. It is clear that this is an optimal ear-decomposition of G , the first ear is even and it contains e_1 and e_2 , so e_1 and e_2 belong to the same block of $\mathcal{M}(G)$.

i.), ii.) and iii.) imply the theorem. □

The following result generalizes Theorem (4.2) and gives some information about the structure of $D(G)$ for an arbitrary 2-edge-connected graph G .

Theorem 6. Let us denote by $G_1, \dots, G_k$ the blocks of $D(G)$. Then

a.) the graph $S(G) := ((G/G_1)/\dots)/G_k$ is factor-critical,

b.) $\varphi(G) = \sum_{i=1}^k \varphi(G_i)$,

c.) $\varphi(G/G_i) = \varphi(G) - \varphi(G_i)$ ($i = 1, \dots, k$),

d.) G_i is φ -covered ($i = 1, \dots, k$).

Proof. We prove the theorem by induction on $\varphi(G)$. First suppose that $\varphi(G) = 1$. Then Theorem (4.2) proves a.), and Theorem (4.2) and Fact ($\times$) in the Introduction implies b.), c.) and d.).

Now suppose that $\varphi(G) \geq 2$. Let H be a strong subgraph of G with strong barrier X . By Theorem 3/b, H is almost critical, and by Theorem (4.2), $B(H)$ is matching-covered. Since $B(H)$ is 2-vertex-connected, by Lemma 1, it is included in some G_i , say G_1 . Consider the graph $G' := G/B(H)$. Since $H/B(H)$ is factor-critical by Theorem (4.2), $\varphi(G') = \varphi(G/H)$ and $E(D(G/H)) = E(D(G'))$. By Lemma 1, $E(D(G)) - E(D(H)) = E(D(G/H))$, so $E(D(G')) = E(D(G)) - E(D(H))$. Thus the blocks $G'_1, \dots, G'_l$ of $D(G')$ are exactly the blocks of $G_1/B(H)$ and $G_2, \dots, G_k$. By Theorem 3/b, $\varphi(G') = \varphi(G/H) = \varphi(G) - 1$, thus, by the induction hypothesis, the theorem is true for G' .

Proposition 3. $B(H)$ is a strong subgraph of G_1 .

Proof. For an edge $e \in E(H)$, e is φ -extreme in H if and only if e is φ -extreme in G by Lemma 1, thus deleting the non φ -extreme edges of G from H the resulting graph is exactly $D(H)$. The factor-critical components of $H - X$ correspond to odd components of $B(H) - X$ because the graph $B(H)$ is nice in H by Theorem (4.1). Thus X is a barrier of $B(H)$. Let Y be a maximal barrier of $B(H)$ including X . Then, since $B(H)$ is matching-covered by Theorem (4.2), Y is a strong barrier of $B(H)$. Since X separates the factor-critical components of $H - X$ from $G - V(H)$ in G , Y separates $B(H) - Y$ from $G_1 - V(B(H))$. It follows that $B(H)$ is a strong subgraph of G_1 with strong barrier Y . $\square$

a.) Since $S(G) = S(G')$ (in the second case we contracted G_1 in two steps, namely first $B(H)$ and then the blocks of $G_1/B(H)$), the statement follows from the induction hypothesis.

b.) By Proposition 3 and Theorem 3/b, $\varphi(G_1/B(H)) = \varphi(G_1) - 1$. $\varphi(G) = \varphi(G') + 1 = \sum_1^l \varphi(G'_i) + 1 = \varphi(G_1) - 1 + \sum_2^k \varphi(G_i) + 1 = \sum_1^k \varphi(G_i)$.

c.) By Theorem 1/b, $\varphi(G) \leq \varphi(G/G_i) + \varphi(G_i)$ and $\varphi(G/G_i) \leq \varphi((G/G_i)/\cup_{j \neq i} G_j) + \sum_{j \neq i} \varphi(G_j)$. By adding them, and using that $\varphi((G/G_i)/\cup_{j \neq i} G_j) = 0$ by a.), and $\sum_1^k \varphi(G_j) = \varphi(G)$ by b.), we have the following. $\varphi(G) \leq \sum_1^k \varphi(G_j) = \varphi(G)$. Thus equality holds everywhere, hence $\varphi(G) = \varphi(G/G_i) + \varphi(G_i)$, as we claimed.

d.) For $i \geq 2$ the statement follows from the induction hypothesis. For G_1 it follows from the induction hypothesis, Proposition 3 and from Lemma 1. $\square$

Theorem 7. The edge sets of the blocks of $D(G)$ and the blocks of $\mathcal{M}(G)$ coincide.

Proof. (a) Let e and f be two edges of G so that they belong to the same block of $\mathcal{M}(G)$. By Lemma 2, there exists an optimal ear-decomposition of G so that the starting even ear C contains e and f . Since every edge of C is φ -extreme, the edges of this cycle belong to the same block of $D(G)$.

(b) Let e and f be two edges of G so that they belong to the same block of $D(G)$, say to G_1 . By Theorem 6/d, G_1 is φ -covered, thus, by Theorem 5 and Lemma 2, there exists an optimal ear-decomposition of G_1 so that the starting even ear contains e and f . By Theorem 6/c, $\varphi(G/G_1) = \varphi(G) - \varphi(G_1)$, so this ear-decomposition can be extended to an optimal ear-decomposition of G so that the starting even ear contains e and f . Then by Lemma 2, e and f belong to the same block of $\mathcal{M}(G)$. $\square$

4 The Tight Cut Lemma

We consider only simple graphs in this section. A graph G is called *bicritical* if for each pair of vertices u and v of G the graph $G - u - v$ has a perfect matching, in other words, it is matching-covered and saturated. A *brick* is a 3-vertex-connected bicritical graph.

Proposition 4. [6] *Let G be a graph with a perfect matching. Then G is bicritical if and only if G contains no barrier of size at least two.* $\square$

For a subset S of vertices, $\delta(S)$ denotes the set of edges leaving S . $\delta(S)$ is a *cut*, and a cut $\delta(S)$ is called *odd* if $|S|$ is odd. An odd cut $\delta(S)$ is *trivial* if $|S| = 1$ or $|\bar{S}| = 1$, otherwise *non-trivial*. An odd cut is called *tight* if it contains exactly one edge of each perfect matching. For a cut $\delta(S)$, a perfect matching M is called *S -fat* if $|\delta(S) \cap M| \geq 2$. Note that if S is an odd cut and M is an S -fat perfect matching, then $|\delta(S) \cap M| \geq 3$.

The aim of this section is to present a simple proof for the so-called Tight Cut Lemma of Edmonds, Lovász, Pulleyblank [2].

Theorem 8. *Let $\delta(S)$ be a non-trivial odd cut in a brick G . Then there exists an S -fat perfect matching in G . In other words, a brick does not contain non-trivial tight cuts.*

Proof. Suppose there exists a non-trivial tight cut $\delta(S)$. Then $G[S], G[\bar{S}]$ are connected, otherwise an S -fat perfect matching can easily be found. Let us choose S so that $\delta(S)$ is a non-trivial tight cut and $|S|$ is minimal.

Two cases will be distinguished. Either there exists a vertex $v \in V(G)$ so that $|\delta(v) \cap \delta(S)| \geq 2$ or for each edge uv of G , $\delta(v) \cap \delta(S) = u$ and $\delta(u) \cap \delta(S) = v$. In the first case we will not use the minimality of S thus we may suppose that $v \in \bar{S}$.

CASE 1. Since $G[\bar{S}]$ is connected, v has at least one neighbor in $\bar{S}$. Let us denote the edges incident to v not leaving $\bar{S}$ by $e_1 = vu_1, \dots, e_k = vu_k$. Then by assumption, $1 \leq k \leq d(v) - 2$. Let $G' = G - e_1 - \dots - e_k$. The following lemma contains the main steps in Case 1.

Lemma 3. *Let v be an arbitrary vertex of a brick G . Let $e_i = vu_i$ $1 \leq i \leq k$ be $1 \leq k \leq d(v) - 2$ edges incident to v . Let $G' := G - e_1 - \dots - e_k$. Then*

(3.1) *G' is almost critical and $v \in V(B(G'))$.*

(3.2) *For some $1 \leq i \leq k, u_i \in V(B(G'))$.*

Proof. $G - v$ is factor-critical because G is bicritical. It follows that each edge of G' incident to v is critical making. Thus G' is almost critical and $v \in V(B(G'))$.

We prove (3.2) by induction on k . If $k = 1$, then we may change the role of v and u_1 , and the statement follows from (3.1). Suppose (3.2) is true for each $1 \leq l \leq k - 1$ but not for k . By (3.1) and Theorem (4.3), it follows that there exists a strong subgraph H of G' with strong barrier X so that u_k belongs to an even component C of $G' - X$.

Proposition 5. v is in an odd component F_1 of $G' - X$ and $|X| \neq 1$.

Proof. If $v \in X$, then X is a barrier in G , so by Proposition 4, $X = v$ and since G is 3-vertex-connected there is no even component of $G - X$, a contradiction. If v was in an even component of $G' - X$, then $X \cup v$ is a barrier in G' , and also in G which contradicts Proposition 4. Thus v is in an odd component F_1 of $G' - X$.

If $|X| = 1$, then, since v has at least two neighbors in G' , v has a neighbor in F_1 , thus $|V(F_1)| \geq 3$. It follows that $G - (X \cup v)$ has at least two connected components (C and $F_1 - v$). This is a contradiction because $|X \cup v| = 2$ and G is a brick. $\square$

Proposition 6. Let $R := \{u_i : u_i \in V(H) - X - V(F_1)\}$ and let T be the union of those connected components of $N(G')$ which contain the vertices in R . Then (6.1) $1 \leq |R| \leq k - 1$ and (6.2) $T \subseteq V(H) - X - V(F_1)$.

Proof. There exists $1 \leq j \leq k - 1$ so that u_j belongs to an odd component of $H - X$ different from F_1 , because by Proposition 5 and Proposition 4, X is not a barrier in G . Since by assumption $|R| \leq k - 1$, (6.1) is proved.

(6.2) follows from the facts that each connected component of T contains a vertex in $V(H) - X - V(F_1)$ (some u_i), $T \cap X = \emptyset$ (since by assumption and by Theorem (4.1), $T \cap V(B(G')) = \emptyset$ and by Theorem 4.6/a in [3], $X \subseteq V(B(G'))$), and X is a cutset in G' . $\square$

Let $G'' := G - \{e_i : u_i \in R\}$. Using (6.1), the induction hypothesis implies that there exists $u_j \in R$ so that $u_j \in V(B(G''))$.

Proposition 7. There exists an allowed edge of G' leaving T .

Proof. There exists an allowed edge $f = st$ of G'' so that $t \in T$, $s \notin T$ by Theorem (4.4) using that $v \notin T$ (since $v \in V(B(G'))$ by (3.2) and $T \cap V(B(G')) = \emptyset$), $v \in V(B(G''))$ by (3.1) and $u_j \in T \cap V(B(G''))$. Let M_1 be a perfect matching of G'' which contains f .

To prove the proposition we show that f is an allowed edge of G' . There exists an odd component $F^* \neq F_1$ of $G' - X$ which contains t by (6.2). By adding some edges to G' between F_1 and some even components of $G' - X$, the odd components of $G' - X$ different from F_1 does not change. Consequently, X is a barrier in G'' and F^* is an odd component of $G'' - X$. Thus there exists exactly one edge g of M_1 leaving F^* . g is an allowed edge in G' by Theorem

4.6/b in [3], Lemma 2.6 in [7] and Theorem (4.1). Let M_2 be a perfect matching of G' containing g . g is the only edge of M_2 leaving F^* because X is a barrier in G' . Thus $M_1(F^*) \cup M_2(G' - V(F^*)) \cup g$ is a perfect matching of G' containing f . $\square$

Proposition 7 gives a contradiction because, by the definition of T , there is no allowed edge leaving T in G' . $\square$

Let $T := S \cup v$. By (3.1) G' is almost critical and $v \in V(B(G'))$, by (3.2) for some neighbor u_i of v , $u_i \in V(B(G'))$, thus by Theorem (4.4), $\delta(T)$ contains an allowed edge f of G' . Let M be a perfect matching of G' containing f . Since v has no neighbor in $\overline{S}$ in G' M matches v to a vertex in S . Hence M is an S -fat perfect matching in G , and we are done in Case 1.

CASE 2. The following proposition is the only place where we need the minimality of S .

Proposition 8. $G[S - u]$ is connected for each edge uv with $u \in S, v \in \overline{S}$.

Proof. Suppose not, that is there exists an edge uv with $u \in S, v \in \overline{S}$ and a partition of $S - u$ into two sets $S_1 \neq \emptyset \neq S_2$ so that there is no edge between S_1 and S_2 . First suppose that $|S_1|$ and $|S_2|$ are odd. Then an arbitrary perfect matching of G containing uv is S -fat. Now suppose that $|S_1|$ and $|S_2|$ are even. Then $S' := S_1 \cup v$ is a non-trivial odd cut and $|S'| < |S|$. Then, by the minimality of S , there exists an S' -fat perfect matching M of G . Since $|\delta(S') \cap M| \geq 3$ and there is no edge between S_1 and S_2 , M is also S -fat. $\square$

Proposition 9. $G[\overline{S} - v]$ is connected for some edge uv with $u \in S, v \in \overline{S}$.

Proof. Let us consider the blocks of $G[\overline{S}]$. If it has only one block, then $G[\overline{S} - v]$ is connected for each vertex. If it has more blocks, then, because of the tree structure of the blocks, there exists a block B that contains only one cut vertex. G contains no cut vertex so there exists a vertex $v \in B$ so that v has a neighbor u in S and $B - v$ is connected. In both cases the lemma is proven. $\square$

By Proposition 9, by Proposition 8 and since we are in Case 2, there exists an edge uv so that $u \in S, v \in \overline{S}$, v has only one neighbor in S (namely u) and u has only one neighbor in $\overline{S}$ (namely v), and $G[S - u]$ and $G[\overline{S} - v]$ are connected. From now on, the minimality of S will not be used, that is S and $\overline{S}$ play the same role.

Let $G'' := G - u - v$. Clearly,

(**) G'' has a perfect matching but there exists no allowed edge of G'' between $S - u$ and $\overline{S} - v$.

Proposition 10. Let H be a strong subgraph of G'' with strong barrier X . Then

(10.1) $(S - u) \cap V(H) \neq \emptyset$ and $(\overline{S} - v) \cap V(H) \neq \emptyset$.

(10.2) Either $X \subseteq (S - u)$ or $X \subseteq (\overline{S} - v)$.

Proof. If say $(S - u) \cap V(H) = \emptyset$, then $X \cup v$ is a barrier of G because all neighbors of u are in $S \cup v$. Since $|X \cup v| \geq 2$ this contradicts Proposition 4.

By Theorem 3/b, H is almost critical. Thus, by Theorem 4.3/b in [3], $X \subseteq V(B(H))$, so by Theorem (4.1) X belongs to a connected component of $N(H)$. By (**), G'' has a perfect matching thus X belongs to a connected component of $N(G'')$. Then, by (**), X is disjoint either from $(S - u)$ or from $\bar{S} - v$, which was to be proved. $\square$

Proposition 11. *G'' is almost critical.*

Proof. By (**), G'' has a perfect matching. If G'' is not almost critical, then by Corollary 4.8 in [3], there exist two disjoint strong subgraphs H_1 and H_2 in G'' with strong barriers X_1 and X_2 . By Proposition (10.2), we may suppose that $X_1 \subseteq (\bar{S} - v)$. By Proposition (10.1), $(S - u) \cap V(H_1) \neq \emptyset$, thus $(S - u) \subseteq V(H_1)$ because $G[S - u]$ is connected. Hence $(S - u) \cap V(H_2) = \emptyset$, contradicting Proposition (10.1). $\square$

Proposition 12. *$(S - u) \cap V(B(G'')) \neq \emptyset$ and $(\bar{S} - v) \cap V(B(G'')) \neq \emptyset$.*

Proof. Suppose $(S - u) \cap V(B(G'')) = \emptyset$. We claim that there exists a strong subgraph H of G'' such that $(S - u) \cap V(H) = \emptyset$. This will give a contradiction because of Proposition (10.1). Let $w \in S - u$. By Proposition 11, G'' is almost critical. Since, by assumption, $w \notin V(B(G''))$, it follows by Theorem (4.3) that there exists a strong subgraph H with strong barrier X so that $w \notin V(H)$, that is w belongs to an even component C of $G'' - X$. By Theorem 4.3/b in [3], $X \subseteq V(B(G''))$, thus $X \cap (S - u) = \emptyset$. Since $G[S - u]$ is connected, $S - u \subseteq C$. $\square$

By Proposition 11, Theorem (4.4) and Proposition 12 there exists an allowed edge f of G'' between $S - u$ and $\bar{S} - v$. This contradicts (**). $\square$

5 φ -Covered Graphs

The aim of this section is to extend earlier results on matching-covered graphs of Lovász and Plummer [6] for φ -covered graphs. First we prove a technical lemma.

Lemma 4. *Let e be an edge of a φ -covered graph G and assume that $\varphi(G) \geq 2$. Then there exists a strong subgraph H of G so that $e \in E(G/H)$.*

Proof. First suppose that G has a perfect matching. Then, by Corollary 4.8 in [3], G has two vertex disjoint strong subgraphs. Clearly, for one of them $e \in E(G/H)$. Secondly, suppose that G has no perfect matching. Let X be a maximal vertex set for which $c_o(G - X) > |X|$. Then each component of $G - X$ is factor-critical and, by assumption, $|X| \neq \emptyset$.

i.) If a component F contains an end vertex of e , then by Lemma 1.8 in [8], G has a strong subgraph H so that $V(H) \subseteq V(G) - V(F)$ so we are done.

ii.) Otherwise, by Lemma 1.8 in [8], G has a strong subgraph H with strong barrier $Y \subseteq X$ so that each component of $H - Y$ is a component of $G - X$. We claim that $e \in E(G/H)$. If not then the two end vertices u and v of e belong to Y because we are in ii.). By Lemma 1, H is φ -covered and e is φ -extreme in H . Thus by Theorem 3/b and Fact ($\times$), H is matching-covered, that is H/e is factor-critical. However, $c_o((H/e) - (Y/e)) = |Y| > |Y/e|$, so obviously, H/e is not factor-critical, a contradiction. $\square$

The following theorem generalizes Theorem 2 for φ -covered graphs. It is equivalent to Theorem 5 by Lemma 2.

Theorem 9. *Let G be a 2-vertex-connected φ -covered graph. Then for every pair of edges $\{e, f\}$ there exists an optimal ear-decomposition of G so that the first ear is even and it contains e and f .* $\square$

Lovász and Plummer [6] proved that each matching-covered graph has a 2-graded ear-decomposition. This result may also be generalized for φ -covered graphs. A sequence $(G_0, G_1, \dots, G_m)$ of subgraphs of G is a *generalized 2-graded ear-decomposition* of G if G_0 is a vertex, $G_m = G$, for every $i = 0, \dots, m-1$: G_{i+1} is φ -covered, G_{i+1} is obtained from G_i by adding at most two disjoint paths (ears) which are openly disjoint from G_i but their end-vertices belong to G_i , if we add two ears then both are of odd length, and $\varphi(G_i) \leq \varphi(G_{i+1})$. This is the natural extension of the original definition of Lovász and Plummer. Indeed, if G is matching-covered then $\varphi(G) = 1$, thus the first ear will be even and all the other ears will be odd.

Theorem 10. *Let e be an arbitrary edge of a φ -covered graph G . Then G has a generalized 2-graded ear-decomposition so that the starting ear contains e .*

Proof. We prove the theorem by induction on $|V(G)|$. Let H be a strong subgraph of G with strong barrier X . By Lemma 1, H and G/H are φ -covered. By Theorem 3/b and Fact ($\times$), H is matching-covered. Let us denote by v the vertex of G/H corresponding to H .

Case 1. $e \in E(H)$. By the result of Lovász and Plummer [6], H has a 2-graded ear-decomposition $(G_0, G_1, \dots, G_k)$ so that the starting ear contains e . Let e' be an edge of G/H incident to v . By induction G/H has a 2-graded ear-decomposition $(G'_0, G'_1, \dots, G'_l)$ so that the starting ear contains e' . Let $G''_i = G_i$ if $0 \leq i \leq k$ and let G''_i be the graph obtained from G'_{i-k-1} by replacing the vertex v by G_k if $k+1 \leq i \leq k+l+1$. We show that $(G''_0, G''_1, \dots, G''_{k+l+1})$ is the desired generalized 2-graded ear-decomposition. The starting ear contains e , in each step we added at most two ears, when two ears were added then they were of odd length, $\varphi(G''_i) \leq \varphi(G''_{i+1})$ and finally by Lemma 1, each subgraph G''_i is φ -covered.

Case 2. $e \notin E(H)$. Let us denote by Q the block of G/H which contains e . Since G/H is φ -covered, Q is also φ -covered. By induction Q has a generalized 2-graded ear-decomposition $(G_0, G_1, \dots, G_k)$ so that the starting ear contains e .

Let G_j be the first subgraph of Q which contains v and let a and b the two edges of G_j incident to v . $G/H/Q$ is also φ -covered, since a graph is φ -covered if and only if each block of it is φ -covered. By induction $G/H/Q$ has a generalized 2-graded ear-decomposition $(G_0^*, G_1^*, \dots, G_p^*)$ so that the starting ear contains an edge incident to v .

i.) First suppose that a and b are incident to the same vertex u of X in G . Let c be an edge of H incident to u . H has a 2-graded ear-decomposition $(G'_0, G'_1, \dots, G'_l)$ so that the starting ear contains c . Let $G''_i = G_i$ if $0 \leq i \leq j$, let G''_i be the graph obtained from G'_{i-j-1} by replacing the vertex u by G_j if $j+1 \leq i \leq j+l+1$, let G''_i be the graph obtained from G_{i-l-1} by replacing the vertex v by G''_{j+l+1} if $j+l+2 \leq i \leq k+l+1$ and finally let G''_i be the graph obtained from $G_{i-k-l-1}$ by replacing the vertex v by G''_{k+l+1} if $k+l+2 \leq i \leq k+l+p+1$. As above, it is easy to see that $(G''_0, G''_1, \dots, G''_{k+l+p+1})$ is the desired generalized 2-graded ear-decomposition.

ii.) Secondly, suppose that a and b are incident to different vertices u and w of X in G . Let c and d be edges of H incident to u and w respectively. By Theorem 5.4.2 in [6] and Theorem 2, H has a 2-graded ear-decomposition $(G'_0, G'_1, \dots, G'_l)$ so that the starting ear P_1 contains c and d . u and w divide P_1 (which is an even ear) into two paths D_1 and D_2 . By Proposition 2, D_1 and D_2 are of even length. Let $G''_i = G_i$ if $0 \leq i \leq j-1$, G''_j be graph obtained from G_j by replacing the vertex v by D_1 , G''_{j+1} be graph obtained from G_j by replacing the vertex v by P_1 , let G''_i be graph obtained from G'_{i-j-1} by replacing P_1 by G''_{j+1} if $j+2 \leq i \leq j+l+1$, let G''_i be graph obtained from G_{i-l-1} by replacing the vertex v by G''_{j+l+1} if $j+l+2 \leq i \leq k+l+1$ and finally as above let G''_i be the graph obtained from $G_{i-k-l-1}$ by replacing the vertex v by G''_{k+l+1} if $k+l+2 \leq i \leq k+l+p+1$. It is easy to see that $(G''_0, G''_1, \dots, G''_{k+l+p+1})$ is the desired generalized 2-graded ear-decomposition. In this case however in order to see that each subgraph G''_i ($j+2 \leq i \leq j+l+1$) is φ -covered we have to use that the subgraph of G''_i corresponding to G'_{i-j-1} of H constructed so far is a strong subgraph of G''_i . $\square$

The two ear theorem on matching-covered graphs follows easily from the following: if the addition of some new edge set F to a matching-covered graph G results in a matching-covered graph then we can choose at most two edges of F so that by adding them to G the graph obtained is matching-covered. The next theorem is the natural generalization of this. However, we cannot prove Theorem 10 using this result.

Theorem 11. *Let $F := \{e_1, \dots, e_k\} \subset E(\overline{G})$ be a set of non edges of a φ -covered graph G . Suppose that $G + F$ is φ -covered and $\varphi(G) = \varphi(G + F)$. Then there exist $i \leq j$ so that $G + e_i + e_j$ is φ -covered.*

Proof. We prove the theorem by induction on $\varphi(G)$. If $\varphi(G) = 1$, then G is matching-covered by Fact ($\times$), so by the theorem of Lovász and Plummer (Lemma 5.4.5 in [6]), we are done. In the following we suppose that $\varphi(G) \geq 2$. Let $F' \subseteq F$ be a minimal set in F so that $G' := G + F'$ is φ -covered. We claim

that $|F'| \leq 2$. Suppose that $|F'| \geq 3$ and let e be an edge of F' . By Lemma 4, there exists a strong subgraph H of G' so that $e \in E(G'/H)$. By Lemma 1, H and G'/H are φ -covered. Let $E_1 := E(H) \cap F'$ and $E_2 := E(G'/H) \cap F'$. Then $E_1 \cup E_2 = F'$ and $E_2 \neq \emptyset$.

First suppose that $E_1 = \emptyset$. Then H is also a strong subgraph of G , so by Lemma 1, G/H is φ -covered. By Theorem 3, $\varphi(G/H) = \varphi(G) - 1$, thus by induction for G/H and F' , there exists $F'' \subseteq F'$ so that $|F''| \leq 2$ and $(G/H) + F''$ is φ -covered. By Lemma 1, $G + F''$ is φ -covered, and we are done.

Secondly, suppose that $E_1 \neq \emptyset$. We claim that each edge of $G + E_1$ is φ -extreme. Clearly, each edge of G is φ -extreme in $G + E_1$. Furthermore, each edge of E_1 is φ -extreme in H , so by Lemma 1, they are φ -extreme in $G + E_1$. Since $E_1 \subset F'$, this contradicts the minimality of F' . $\square$

Example. We give an example which shows that the above theorem is not true if we delete the condition that $\varphi(G) = \varphi(G + F)$. Let G be the graph obtained from a star on four vertices by duplicating each edge and let F be the three edges in $E(\overline{G})$. Then G and $G + F$ are φ -covered but for every non empty subset F' of F , $G + F'$ is not φ -covered. Note that $\varphi(G) = 3$ and $\varphi(G + F) = 1$.

Acknowledgement This research was done while the author visited the Research Institute for Discrete Mathematics, Lennéstrasse 2, 53113. Bonn, Germany by an Alexander von Humboldt fellowship. All the results presented in this paper can be found in [10] or in [11].

References

1. J. Cheriyan, A. Sebő, Z. Szigeti, An improved approximation algorithm for the minimum 2-edge-connected spanning subgraph problem, Proceedings of IPCO6, eds.: R.E. Bixby, E.A. Boyd, R.Z. Rios-Mercado, (1998) 126-136.
2. J. Edmonds, L. Lovász, W. R. Pulleyblank, Brick decompositions and the matching rank of graphs, *Combinatorica* **2** (1982) 247-274.
3. A. Frank, Conservative weightings and ear-decompositions of graphs, *Combinatorica* **13** (1) (1993) 65-81.
4. C.H.C. Little, A theorem on connected graphs in which every edge belongs to a 1-factor, *J. Austral. Math. Soc.* **18** (1974) 450-452.
5. L. Lovász, A note on factor-critical graphs, *Studia Sci. Math. Hungar.* **7** (1972) 279-280.
6. L. Lovász, M. D. Plummer, "Matching Theory," North Holland, Amsterdam, (1986).
7. Z. Szigeti, On Lovász's Cathedral Theorem, Proceedings of IPCO3, eds.: G. Rinaldi, L.A. Wolsey, (1993) 413-423.
8. Z. Szigeti, On a matroid defined by ear-decompositions of graphs, *Combinatorica* **16** (2) (1996) 233-241.
9. Z. Szigeti, On the two ear theorem of matching-covered graphs, *Journal of Combinatorial Theory, Series B* **74** (1998) 104-109.
10. Z. Szigeti, Perfect matchings versus odd cuts, Technical Report No. 98865 of Research Institute for Discrete Mathematics, Bonn, 1998.
11. Z. Szigeti, On generalizations of matching-covered graphs, Technical Report of Research Institute for Discrete Mathematics, Bonn, 1998.

Gale-Shapley Stable Marriage Problem Revisited: Strategic Issues and Applications (Extended Abstract)

Chung-Piaw Teo^{1*}, Jay Sethuraman^{2**}, and Wee-Peng Tan¹

¹ Department of Decision Sciences, Faculty of Business Administration,
National University of Singapore
fbateocp@nus.edu.sg

² Operations Research Center, MIT, Cambridge, MA 02139
jayc@mit.edu

1 Introduction

This paper is motivated by a study of the mechanism used to assign primary school students in Singapore to secondary schools. The assignment process requires that primary school students submit a rank ordered list of six schools to the Ministry of Education. Students are then assigned to secondary schools based on their preferences, with priority going to those with the highest examination scores on the Primary School Leaving Examination (PSLE). The current matching mechanism is plagued by several problems, and a satisfactory resolution of these problems necessitates the use of a *stable matching* mechanism. In fact, the *student-optimal* and *school-optimal* matching mechanisms of Gale and Shapley [2] are natural candidates.

Stable matching problems were first studied by Gale and Shapley [2]. In a stable marriage problem we have two finite sets of players, conveniently called the set of men (M) and the set of women (W). We assume that every member of each set has strict preferences over the members of the opposite sex. In the *rejection* model, the preference list of a player is allowed to be incomplete in the sense that players have the option of declaring some of the members of the opposite sex as unacceptable; in the *Gale-Shapley* model we assume that preference lists of the players are *complete*. A matching is just a one-to one mapping between the two sexes; in the rejection model, we also include the possibility that a player may be unmatched, i.e. the player's assigned partner in the matching is himself/herself. The matchings of interest to us are those with the crucial *stability* property, defined as follows: A matching μ is said to be *unstable* if there is a man-woman pair, who both prefer each other to their (current) assigned partners in μ ; this pair is said to block the matching μ , and is called a *blocking pair* for μ . A *stable matching* is a matching that is not unstable. The significance of stability is best highlighted by a system where acceptance of the proposed matching is

* Research supported by a NUS Research grant RP 3970021.

** Research supported by an IBM Fellowship.

voluntary. In such a setting, an unstable matching cannot be expected to remain intact, as the blocking pair(s) would soon discover that they could both improve their match by joint action; the man and the woman involved in a blocking pair could just “divorce” their respective partners and “elope.” To put this model in context, the students play the role of “women” and the secondary schools play the role of the “men.” Observe that there is a crucial difference between the stable marriage model as described here, and the problem faced by the primary school students in Singapore; in the latter, many “women” (students) can be assigned to the same “man” (secondary school), whereas in the former we require that the matching be one-to-one. In what follows, we shall restrict our attention to the one-to-one marriage model; nevertheless, the questions studied here, and the ideas involved in their resolution are relevant to the many-to-one marriage model as well.

One of the main difficulties with using the men-optimal stable matching mechanism is that it is manipulable by the women: some women can intentionally misrepresent their preferences to obtain a better stable partner. Such (strategic) questions have been studied for the stable marriage problem by mathematical economists and game theorists; essentially, this approach seeks to understand and quantify the potential gains of a deceitful player. Roth [7] proved that when the men-optimal stable-matching mechanism is used, a man will never be better off by falsifying his preferences, if all the other people in the problem reveal their true preferences. Falsifying preferences will at best result in the (original) match that he obtains when he reveals his true preferences. Gale and Sotomayor [4] showed that when the man-propose algorithm is used, a woman w can still force the algorithm to match her to her women-optimal partner, denoted by $\mu(w)$, by falsifying her preference list. The optimal cheating strategy for woman w is to declare men who rank below $\mu(w)$ as unacceptable. Indeed, the cheating strategy proposed by Gale and Sotomayor is optimal in the sense that it enables the women to obtain their women-optimal stable partner even when the man-propose mechanism is being adopted. Prior to this study, we know of no analogous results when the women are required to submit *complete* preference lists. This question is especially relevant to the Singapore school-admissions problem: All assignments are done via the centralized posting exercise, and no student is allowed to approach the schools privately for admission purposes. In fact, in the current system, students not assigned to a school on their list are assigned to schools according to some pre-determined criterion set by the Ministry. Thus effectively this is a matching system where the students are not allowed to remain “single” at the end of the posting exercise. To understand whether the stable matching mechanism is a viable alternative, we first need to know whether there is any incentive for the students to falsify their preferences, so that they can increase their chances of being assigned to “better” schools. The one-to-one version of this problem is exactly the question studied in this paper: in the stable marriage model with complete preferences, with the men-optimal matching mechanism, is there an incentive for the women to cheat? If so, what is the optimal cheating strategy for a woman? To our knowledge, the only result known about this prob-

lem is an example due to Josh Benaloh (cf. Gusfield and Irving [5]), in which the women lie by permuting their preference lists, and still manage to force the men-optimal matching mechanism to return the women-optimal solution.

2 Optimal Cheating in the Stable Marriage Problem

Before we derive the optimal cheating strategy we consider a (simpler) question: Suppose woman w is allowed to reject proposals. Is it possible for woman w to identify her women-optimal partner by observing the sequence of proposals in the men-propose algorithm? Somewhat surprisingly, the answer is yes! Our algorithm to compute the optimal cheating strategy is motivated by the observation that if a woman simply rejects all the proposals made to her, then the best (according to her true preference list) man among those who have proposed to her is her women-optimal partner. Hence by rejecting all her proposals, a woman can extract information about her best possible partner. Our algorithm for the optimal cheating strategy builds on this insight: the deceitful woman rejects as many proposals as possible, while remaining engaged to some man who proposed earlier in the algorithm. Using a backtracking scheme, the deceitful woman can use the matching mechanism repeatedly to find her optimal cheating strategy.

2.1 Finding Your Optimal Partner

We first describe algorithm OP —an algorithm to compute the women-optimal partner for w using the man-propose mechanism. (Recall that we do this under the assumption that woman w is allowed to remain single.)

Algorithm OP

1. Run the man-propose algorithm, and reject all proposals made to w . At the end, w and a man, say m , will remain single.
2. Among all the men who proposed to w in Step 1, let the best man (according to w) be m_1 .

Theorem 1. m_1 is the women-optimal partner for w .

Proof: Let $\mu(w)$ denote the women-optimal partner for w . We modify w 's preference list by inserting the option to remain single in the list, *immediately after* $\mu(w)$. (We declare all men that are inferior to $\mu(w)$ as unacceptable to w .) Consequently, in the man-propose algorithm, all proposals inferior to $\mu(w)$ will be rejected. Nevertheless, since there exists a stable matching with w matched to $\mu(w)$, our modification does not destroy this solution. It is also well known that the set of people who are single is the same for all stable matchings (cf. Roth and Sotomayor [8], pg. 42). Thus, w must be matched in *all* stable matchings with the modified preference list. The men-optimal matching for this modified preference list must match w to $\mu(w)$. In particular, $\mu(w)$ must have proposed to

w during the execution of the man-propose algorithm. Note that until $\mu(w)$ proposes to w , the man-propose algorithm for the modified list runs exactly in the same way as in Step 1 of OP . The difference is that Step 1 of OP will reject the proposal from $\mu(w)$, while the man-propose algorithm for the modified list will accept the proposal from $\mu(w)$, as w prefers $\mu(w)$ to being single. Hence, clearly $\mu(w)$ is among those who proposed to w in Step 1 of OP , and so $m_1 \geq_w \mu(w)$.

Suppose $m_1 >_w \mu(w)$. Consider the modified list in which we place the option of remaining single *immediately after* m_1 . We run the man-propose algorithm with this modified list. Again, until m_1 proposes to w , the algorithm runs exactly the same as in Step 1 of OP , after which the algorithm returns a stable partner for w who is at least as good as m_1 . This gives rise to a contradiction as we assumed $\mu(w)$ to be the best stable partner for w . ■

Observe that under this approach, the true preference list of w is only used to compare the men who have proposed to w . We do not need to know her exact preference list; we only need to know which man is the best among a given set of men, according to w . Hence the information set needed here to find the women-optimal partner of w is much less than that needed when the woman-propose algorithm is used. This is useful for the construction of the cheating strategy as the information on the “optimal” preference list is not given a-priori and is to be determined.

2.2 Cheating Your Way to a Better Marriage

Observe that the preceding procedure only works when woman w is allowed to remain single throughout the matching process, so that she can reject any proposal made to her in the algorithm. Suppose we do not give the woman an option to declare any man as unacceptable. How do we determine her best stable partner? This is essentially a restatement of our original question: who is the best stable partner woman w can have when the man-propose algorithm is used and when she can lie only by permuting her preference list.

A natural extension of Algorithm OP is for woman w to: (i) accept a proposal first, and then reject all future proposals. (ii) From the list of men who proposed to w but were rejected, find her most preferred partner; repeat the Gale-Shapley algorithm until the stage when this man proposes to her. (iii) Reverse the earlier decision and accept the proposal from this most preferred partner, and continue the Gale-Shapley algorithm by rejecting all future proposals. (iv) Repeat (ii) and (iii) until the woman cannot find a better partner from all other proposals. Unfortunately, this elegant strategy does not always yield the best stable partner a woman can achieve under our model. The reason is that this greedy improvement technique does not allow for the possibility of rejecting the current best partner, in the hope that this rejection will trigger a proposal from a *better* would-be partner. Our algorithm in this paper does precisely that. Let $P(w) = \{m_1, m_2, \dots, m_n\}$ be the true preference list of woman w , and let $P(m, w)$ be a preference list for w that returns m as her men-optimal partner. Our algorithm constructs $P(m, w)$ iteratively, and consists of the following steps:

1. Run the man-propose algorithm with the true preference list $P(w)$ for woman w . Keep track of all men who propose to w . Let the men-optimal partner for w be m , and let $P(m, w)$ be the true preference list $P(w)$.
2. Suppose m_j proposed to w in the Gale-Shapley algorithm. By moving m_j to the front of the list $P(m, w)$, we obtain a preference list for w such that the men-optimal partner will be m_j . Let $P(m_j, w)$ be this altered list. We say that m_j is a *potential* partner for w .
3. Repeat step 2 for every man who proposed to woman w in the algorithm; after this, we say that we have *exhausted* man m , the men-optimal partner obtained with the preference list $P(m, w)$.
4. If a potential partner for w , say man u , has not been exhausted, run the Gale-Shapley algorithm with $P(u, w)$ as the preference list of w . Identify new possible partners and their associated preference lists, and classify man u as exhausted.
5. Repeat Step 4 until all possible partners of w are exhausted. Let N denote the set of all possible (and hence exhausted) partners for w .
6. Among the men in N let m_a be the man woman w prefers most. Then $P(m_a, w)$ is an optimal cheating strategy for w .

The men in the set N at the end of the algorithm have the following crucial properties:

- For each man m in N , there is an associated preference list for w such the Gale-Shapley algorithm returns m as the men-optimal partner for w with this list.
- All other proposals in the course of the Gale-Shapley algorithm come from other men in N . (Otherwise, there will be some possible partners who are not exhausted.)

With each run of the Gale-Shapley algorithm, we exhaust a possible partner, and so we need at most n Gale-Shapley algorithms before termination.

Theorem 2. $\pi = P(m_a, w)$ is an optimal strategy for woman w .

Proof: (by contradiction) We use the convention that $r(m) = k$ if man m is the k^{th} man on woman w 's list. Let $\pi^* = \{m_{p1}, m_{p2}, \dots, m_{pn}\}$ be the preference list that gives rise to the *best* stable partner for w under the man-propose algorithm. Let this man be denoted by m_{pb} , and let woman w strictly prefer m_{pb} to m_a (under her true preference list). Recall that we use $r(m)$ to represent the rank of m under the true preferences of w ; by our assumption, $r(m_{pb}) < r(m_a)$, i.e., m_{pb} is ranked higher than m_a . Observe that the order of the men who do not propose to woman w is irrelevant and does not affect the outcome of the Gale-Shapley's algorithm. Furthermore, men of rank higher than $r(m_{pb})$ do not get to propose to w , otherwise we can cheat further and improve on the best partner for w , contradicting the optimality of π^* . Thus we can arbitrarily alter the order of these men, without affecting the outcome. Without loss of generality, we may assume that $1 = r(m_{p1}) < 2 = r(m_{p2}) < \dots < q = r(m_{pb})$.

Since $r(m_{pb}) < r(m_a)$, m_a will appear anywhere after m_{pb} in π^* : thus, m_a can appear in any position from m_{pb+1} to m_{pn} .

Now, we modify π^* such that all men who (numerically) rank lower than m_a but higher than m_{pb} (under true preferences) are put in order according to their ranks. This is accomplished by moving all these men before m_a in π^* . With that alteration, we obtain a new list $\tilde{\pi} = \{m_{q1}, m_{q2}, \dots, m_{qn}\}$ such that:

- (i) $1 = r(m_{q1}) < 2 = r(m_{q2}) < \dots < s = r(m_{qs})$.
- (ii) $m_{q1} = m_{p1} \dots m_{qb} = m_{pb}$, where the position of those men who rank higher than m_{pb} is unchanged.
- (iii) $r(m_a) = s + 1$, $m_a \in \{m_{qs+1}, m_{qs+2}, \dots, m_{qn}\}$.
- (iv) The men in the set $\{m_{qs+1}, m_{qs+2}, \dots, m_{qn}\}$ retain their relative position with respect to one another under π^* .

Note that the men-optimal partner of w under $\tilde{\pi}$ cannot come from the set $\{m_{qs+1}, m_{qs+2}, \dots, m_{qn}\}$. Otherwise, since the set of men who proposed in the course of the algorithm must come from $\{m_{qs+1}, m_{qs+2}, \dots, m_{qn}\}$, and since the preference list π^* retains the relative order of the men in this set, the same partner would be obtained under π^* . This leads to a contradiction as π^* is supposed to return a better partner for w . Hence, we can see that under $\tilde{\pi}$, we already get a better partner than under π .

Now, since the preference list π returns m_a with $r(m_a) = s + 1$, we may conclude that the set N (obtained from the final stage of the algorithm) does not contain any man of rank smaller than $s + 1$. Thus $N \subseteq \{m_{qs+1}, m_{qs+2}, \dots, m_{qn}\}$. Suppose $m_{qs+1}, m_{qs+2}, \dots, m_{qw}$ do not belong to the set N , and m_{qw+1} is the first man after m_{qs} who belongs to the set N . By construction of N , there exists a permutation $\hat{\pi}$ with m_{qw+1} as the stable partner for w under the men-optimal matching mechanism. Furthermore, all of those who propose to w in the course of the algorithm are in N , and hence they are no better than m_a to w . Furthermore, all proposals come from men in $\{m_{qw+1}, m_{qw+2}, \dots, m_{qn}\}$, since $N \subseteq \{m_{qs+1}, m_{qs+2}, \dots, m_{qn}\}$.

By altering the order of those who did not propose to w , we may assume that $\hat{\pi}$ is of the form $\{m_{q1}, m_{q2}, \dots, m_{qs-1}, m_{qs}, \dots, m_{qw}, m_{qw+1}, \dots\}$, where the first $qw + 1$ men in the list are identical to those in $\tilde{\pi}$. But, the men-optimal stable solution obtained using $\hat{\pi}$ must also be stable under $\tilde{\pi}$, since w is match to m_{qw+1} , and the set of men she strictly prefers to m_{qw+1} is identical in both $\hat{\pi}$ and $\tilde{\pi}$. This is a contradiction as $\tilde{\pi}$ is supposed to return a men-optimal solution better than m_a . Thus π^* does not exist, and so π is optimum and m_a is the best stable partner w can get by permuting her preference list. ■

We now present an example to illustrate how our heuristic works.

Example 1: Consider the following stable marriage problem:

1		2	3	4	5	1	1		1	2	3	5	4
2		3	4	5	1	2	2		2	1	4	5	3
3		5	1	4	2	3	3		3	2	5	1	4
4		3	1	2	4	5	4		4	5	1	2	3
5		1	5	2	3	4	5		5	1	2	3	4

True Preferences of the Men True Preferences of the Women

We construct the optimal cheating strategy for woman 1.

- Step 1: Run Gale-Shapley with the true preference list for woman 1; her men-optimal partner is man 5. Man 4 is the only other man who proposes to her during the Gale-Shapley algorithm. So $P(\text{man5}, \text{woman1}) = (1, 2, 3, 5, 4)$.
- Step 2-3: Man 4 is moved to the head of woman 1's preference list; i.e., $P(\text{man4}, \text{woman1}) = (4, 1, 2, 3, 5)$. Man 5 is exhausted, and man 4 is a potential partner.
- Step 4: As man 4 is not yet exhausted, we run the Gale-Shapley algorithm with $P(\text{man4}, \text{woman1})$ as the preference list for woman 1. Man 4 will be exhausted after this, and man 3 is identified as a new possible partner, with $P(\text{man3}, \text{woman1}) = (3, 4, 1, 2, 5)$.
- Repeat Step 4: As man 3 is not yet exhausted, we run Gale-Shapley with $P(\text{man3}, \text{woman1})$ as the preference list for woman 1. Man 3 will be exhausted after this. No new possible partner is found, and so the algorithm terminates.

Example 1 shows that woman 1 could cheat and get a partner better than the men-optimal solution. However, her women-optimal partner in this case turns out to be man 1. Hence Example 1 also shows that woman 1 cannot always assure herself of getting the women-optimal partner through cheating, in contrast to the case when rejection is allowed in the cheating strategy.

3 Strategic Issues in the Gale-Shapley Problem

By requiring the women to submit complete preference lists, we are clearly restricting their strategic options, and thus many of the strong structural results known for the model with rejection may not hold in this model. This is good news, for it reduces the incentive for a woman to cheat. In the rest of this section, we present some examples to show that the strategic behaviour of the women can be very different under the models with and without rejection.

3.1 The Best Possible Partners (Obtained from Cheating) May Not Be Women-Optimal

In the two-sided matching model with rejection, it is not difficult to see that the women can always force the man-propose algorithm to return the women-optimal

solution (e.g. each woman rejects all those who are inferior to her women-optimal partner). In our model, where rejection is forbidden, the influence of the women is far less, even if they collude. A simple example is when each woman is ranked first by exactly one man. In this case, there is no conflict among the men, and in the men-optimal solution, each man is matched to the woman he ranks first. (This situation arises whenever each man ranks his men-optimal partner as his first choice.) In this case, the algorithm will terminate with the men-optimal solution, regardless of how the women rank the men in their lists. So ruling out the strategic option of remaining single for the women significantly affects their ability to change the outcome of the game by cheating.

By repeating the above analysis for all the other women in Example 1, we conclude that the best possible partner for woman 1, 2, 3, 4, and 5 are respectively man 3, 1, 2, 4, and 3. An interesting observation is that woman 5 cannot benefit by cheating alone (she can only get her men-optimal partner no matter how she cheats). However, if woman 1 cheats using the preference list (3, 4, 1, 2, 5), woman 5 will also benefit by being matched to man 5, who is first in her list.

3.2 Multiple Strategic Equilibria

Suppose each woman w announces a preference list $P(w)$. The set of strategies $\{\pi(1), \pi(2), \dots, \pi(n)\}$ is said to be in strategic equilibrium if none of the women has an incentive to deviate unilaterally from this announced strategy. It is easy to see that if a woman benefits from announcing a different permutation list (instead of her true preference list), then every other woman would also benefit, i.e. every other woman will get a partner who is at least as good as her men-optimal partner (cf. Roth and Sotomayor [8]).

Theorem 3. *If a single woman can benefit by cheating, then the game has multiple strategic equilibria.*

Proof: A strategic equilibrium can be constructed by repeating the proposed cheating algorithm iteratively, improving the partner for some woman at each iteration. (Notice that the partner of a woman at the end of iteration j is at least as good as her partner at the beginning of the iteration j .) The algorithm will thus terminate at a strategic equilibrium, where at least one woman will be matched to someone whom she (strictly) prefers to her men-optimal partner. Another strategic equilibrium is obtained if each woman w announces a list of the form $\{m_1, m_2, \dots, m_n\}$, with m_1 being her men-optimal partner and $m_2, m_3, \dots, m_n$ in the same (relative) order as in her true preference list. Clearly the man-propose algorithm will match woman w to m_1 , since moving m_1 to the front of w 's preference list does not affect the sequence of proposals in the man-propose algorithm. No woman can benefit from cheating, as all other women are already matched to their announced first-ranked partner. Thus we have constructed two strategic equilibria. ■

3.3 Does It Pay To Cheat?

Roth [7] shows that under the man-propose mechanism, the men have no incentives to alter their true preference lists. In the rejection model, however, Gale and Sotomayor [3] show that a woman has an incentive to cheat as long as she has at least two distinct stable partners. Pittel [6] shows that the average number of stable solutions is asymptotic to $n\log(n)/e$, and with high probability, the rank of the women-optimal and men-optimal partner for the woman are respectively $\log(n)$ and $n/\log(n)$. Thus in typical instances of the stable marriage game under the rejection model, most of the women will not reveal their true preference lists.

Many researchers have argued that the troubling implications from these studies are not relevant in practical stable marriage game, as the model assumes that the women have full knowledge of each individual's preference list and the set of all the players in the game. For the model we consider, it is natural to ask whether it pays (as in the rejection model) for a typical woman to solicit information about the preferences of all other participants in the game. We run the cheating algorithm on 1000 instances, generated uniformly at random, for $n = 8$ and the number of women who benefited from cheating is tabulated in Table 1.

Number of Women who benefited	0	1	2	3	4	5	6	7	8
Number of observations	740	151	82	19	7	1	0	0	0

Table 1

Interestingly, the number of women who can gain from cheating is surprisingly low. In fact, in 74% of the instances, the men-optimal solution is their only option, no matter how they cheat. The average percentage of women who benefit from cheating is merely 5.06%.

To look at the typical strategic behaviour on larger instances of the stable marriage problem, we run the heuristic on 1000 random instances for $n = 100$. The cumulative plot is shown in Figure 1. In particular, in more than 60% of the instances at most 10 women (out of 100) benefited from cheating, and in more than 96% of the instances at most 20 women benefited from cheating. The average number of women who benefited from cheating is 9.52%. Thus, the chances that a typical woman can benefit from acquiring complete information (i.e., knowing the preferences of the other players) is pretty slim in our model.

We have repeated the above experiment for large instances of the Gale-Shapley model. Due to computational requirements, we can only run the experiment on 100 random instances of the problem with 500 men and women. Again the insights obtained from the 100 by 100 cases carry over: the number of women who benefited from cheating is again not more than 10% of the total number of the women involved. In fact, the average was close to 6% of the women population in the problem. This suggests that the number of women who can benefit from cheating in the Gale-Shapley model with n women grows at a rate which is slower than a linear function of n . However, detailed probabilistic

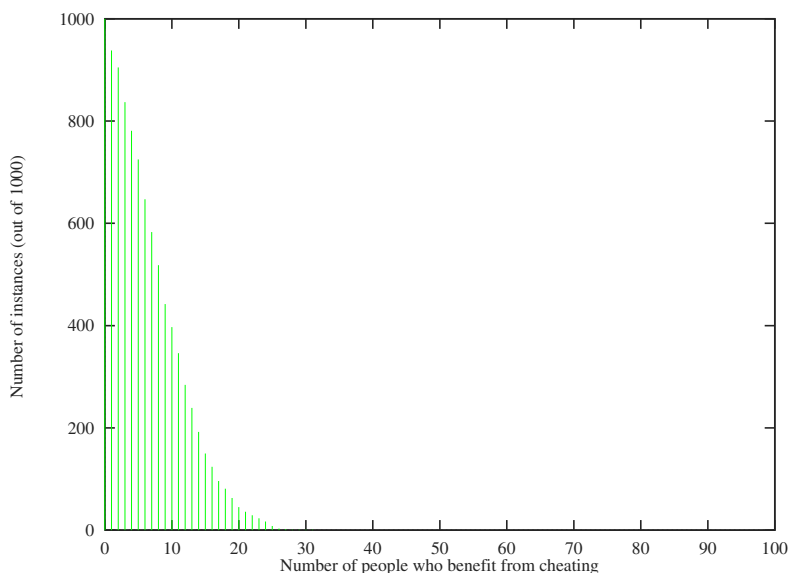


Fig. 1. Benefits of cheating: cumulative plot for $n = 100$

analysis of this phenomenon is a challenging problem that is beyond the scope of the present paper.

A practical advice for all women in the stable marriage game, using men-optimal matching mechanism: “don’t bother to cheat !”

References

1. Dubins, L.E. and D.A. Freedman (1981), “Machiavelli and the Gale- Shapley Algorithm.” *American Mathematical Monthly*, 88, pp485-494.
2. Gale, D., L. S. Shapley (1962). *College Admissions and the Stability of Marriage*. *American Mathematical Monthly* 69 9-15.
3. Gale, D., M. Sotomayor (1985a). *Some Remarks on the Stable Matching Problem*. *Discrete Applied Mathematics* 11 223-232.
4. Gale, D., M. Sotomayor (1985b). *Ms Machiavelli and the Stable Matching Problem*. *American Mathematical Monthly* 92 261-268.
5. Gusfield, D., R. W. Irving (1989). *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Massachusetts.
6. Pittel B. (1989), *the Average Number of Stable Matchings*, *SIAM Journal of Discrete Mathematics*, 530-549.
7. Roth, A.E. (1982). *The Economics of Matching: Stability and Incentives*. *Mathematics of Operations Research* 7 617-628.
8. Roth, A.E., M. Sotomayor (1991). *Two-sided Matching: A Study in Game-theoretic Modelling and Analysis*, Cambridge University Press, Cambridge.

Vertex-Disjoint Packing of Two Steiner Trees: Polyhedra and Branch-and-Cut

Eduardo Uchoa and Marcus Poggi de Aragão

Departamento de Informática - Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 - Rio de Janeiro RJ, 22453-900, Brasil.
uchoa@inf.puc-rio.br, poggi@inf.puc-rio.br

Abstract. Consider the problem of routing the electrical connections among two large terminal sets in circuit layout. A realistic model for this problem is given by the vertex-disjoint packing of two Steiner trees (2VPST), which is known to be NP-complete. This work presents an investigation on the 2VPST polyhedra. The main idea is to depart from facet-defining inequalities for a vertex-weighted Steiner tree polyhedra. Some of these inequalities are proven to also define facets for the packing polyhedra, while others are lifted to derive new important families of inequalities, including proven facets. Separation algorithms are also presented. The resulting branch-and-cut code has an excellent performance and is capable of solving problems on grid graphs with up to 10000 vertices and 5000 terminals in a few minutes.

1 Introduction

The Vertex-disjoint Packing of Steiner Trees problem (VPST) is the following: given a connected graph $G = (V, E)$ and K disjoint vertex-sets $N_1, \dots, N_K$, called *terminal sets*, find disjoint vertex-sets $T_1, \dots, T_K$ such that, for each $k \in \{1, \dots, K\}$, $N_k \subseteq T_k$ and $G[T_k]$ (the subgraph induced by T_k) is connected. Deciding whether a VPST instance is feasible or not is NP-complete, and many times a very challenging task. VPST has a wide application in layout of printed boards or chips, as it can model in an accurate way the routing of electrical connections among the circuit components (see Korte et al. [3] or Lengauer [4]).

This work studies the case where $K = 2$, the Vertex-Disjoint Packing of Two Steiner Trees (2VPST). This restricted problem is still NP-complete [3]. Although most VPST applications involves more than two terminal sets, many problems in circuit layout have only two large terminal sets, for instance the ground and the V_{cc} terminals. Finding a solution for these two large sets may represent a major step in solving the whole routing problem. In fact, it is usual in chip layout to route the ground and V_{cc} connections over special layers, separated from the remaining connections. This particular routing problem leads to the 2VPST.

We address an optimization version of the 2VPST: find two disjoint vertex-sets T_1 and T_2 , such that $G[T_1]$ and $G[T_2]$ are connected, minimizing $|N_1 \setminus T_1| + |N_2 \setminus T_2|$. In other words, we look for a solution leaving the minimum possible

number of unconnected terminals. It is also allowed to set weights in order to specify which terminals have priority in being connected. These features may be very useful for a practitioner and suggests an iterative approach for layout: if the 2VPST instance is not feasible, a best possible solution gives valuable information on how to make small changes in terminal placement to achieve feasibility.

A previous work that is somehow related to ours is Grötschel et al. [1,2], where Edge-disjoint Packing of Steiner Trees (EPST) is studied. EPST gives an alternative model for the routing problem in circuit layout. Their branch-and-cut code could solve switchbox instances (G is a grid graph and all terminals are placed in the outer face) with up to 345 vertices, 24 terminal sets, and 68 terminals.

2 Formulation

For each vertex $i \in V$ define a binary variable x_i , where $x_i = 1$ if i belongs to T_1 and $x_i = 0$ if i belongs to T_2 . This choice of variables obliges every vertex to belong either to T_1 or T_2 . Of course, terminals of N_1 in T_2 and terminals of N_2 in T_1 are considered unconnected. A formulation for 2VPST follows.

$$(F) \left\{ \begin{array}{ll} \text{Min } z = - \sum_{i \in N_1} w_i + \sum_{i \in V} w_i x_i & (1) \\ \text{s.t.} & \\ \sum_{i \in S} x_i - \sum_{i \in R} x_i \leq 1 & \forall S \subseteq V, |S| = 2; \forall R \in \Delta(S) \quad (2) \\ - \sum_{i \in S} x_i + \sum_{i \in R} x_i \leq |R| - 1 & \forall S \subseteq V, |S| = 2; \forall R \in \Delta(S) \quad (3) \\ x_i \in \{0, 1\} & \forall i \in V \quad (4) \end{array} \right.$$

Weight w_i must be negative if $i \in N_1$, positive if $i \in N_2$ and 0 otherwise. The minimum possible number of unconnected terminals matches z^* when $w_i = -1$ for all $i \in N_1$ and $w_i = 1$ for all $i \in N_2$. In any case the 2VPST instance is feasible iff $z^* = 0$. The symbol $\Delta(S)$ denotes the set of all minimal separators of S . A vertex-set R is a *separator* of S if any path joining the pair of vertices in S has at least one vertex in R . Constraints F.2 says: if both vertices in S are in T_1 , then at least one vertex in any separator of S must also be in T_1 . Constraints F.3 are complementary to F.2: if both vertices in S are in T_2 , then at least one vertex in any separator of S must be in T_2 .

3 Polyhedral Investigations

Many combinatorial optimization problems can be naturally decomposed into simpler subproblems. The 2VPST decompose into two Steiner-like subproblems. There are cases where a good polyhedral description for the subproblems yields a good description of the main polyhedra. On the other hand, sometimes even

a complete subproblem description yields but a poor description of the main polyhedra. Experiences with column generation showed that this is indeed the case of 2VPST [6]. In these situations, there is a need to also look for *joint inequalities*, i.e. inequalities which are not valid for any subproblem polyhedra but valid for the main polyhedra.

Definition 1. Given a graph $G = (V, E)$, a v-tree of G is a vertex-set $T \subseteq V$ such that $G[T]$ is connected. A complementary v-tree of G is a vertex-set $T \subseteq V$ such that $G[V \setminus T]$ is connected. A double v-tree of G is a v-tree T of G that is also a complementary v-tree of G . (G being clear, we say v-tree or double v-tree for short).

Definition 2. Let χ^U be the incidence vector of a vertex-set U . Define $P(G)$ as the convex hull of vectors χ^T such that T is a double v-tree of G . Define $P_1(G)$ as the convex hull of vectors χ^T such that T is a v-tree of G and $P_2(G)$ as the convex hull of vectors χ^T such that T is a complementary v-tree of G .

The vectors defining polyhedra P correspond to the solutions of formulation F, so P is associated to 2VPST. Polyhedra P_1 and P_2 are associated to a Vertex Weighted Steiner Tree problem: given a graph with positive and negative vertex weights, find a tree with maximum/minimum total vertex weight. Note that P_1 and P_2 are complementary polyhedra and that every valid inequality for one can be converted into a valid inequality for the other by variable complementation, i.e. replacing x by $1 - x$. For instance, inequalities F.2 are valid for P_1 and the complementary inequalities F.3 are valid for P_2 . Therefore whenever we talk about polyhedra P_1 , it is implicit that similar results also hold for P_2 .

Our idea is to use subproblem polyhedra P_1 and P_2 as a starting point for investigating main polyhedra P . We show that: (i) some inequalities that define facets of P_1 or P_2 also define facets of P and (ii) some of these inequalities which do not define facets of P can be lifted to derive new joint inequalities.

Theorem 1. $P(G)$, $P_1(G)$ and $P_2(G)$ are full-dimensional.

Proof. Let A be a spanning tree of G , rooted at an arbitrary vertex. For each $i \in V$, the vertex-set U_i containing i and his descendents in A is a double v-tree, so $\chi^{U_i} \in P(G)$. As $\chi^\emptyset \in P(G)$, there are $|V| + 1$ affine independent vectors in $P(G)$. As $P(G) \subseteq P_1(G) \cap P_2(G)$, the result follows. $\square$

3.1 The p -Separator Facets of P_1

Definition 3. Let S be a vertex-set, where $|S| \geq 2$. A vertex-set R is a separator of S if any path joining a pair of vertices in S has at least one vertex in R . A separator R of S is proper if $S \cap R = \emptyset$ and minimal if no other separator of S is contained in R . The symbol $\Delta(S)$ indicates the set of all proper and minimal separators of S . Given a separator R of S and a vertex $i \in V$, define S_i^R as the maximum subset of vertices in S that can belong to a v-tree T_i containing i and such that $T_i \cap R \subseteq \{i\}$. Note that a separator R of S is minimal iff $|S_i^R| \geq 2$ for each $i \in R$.

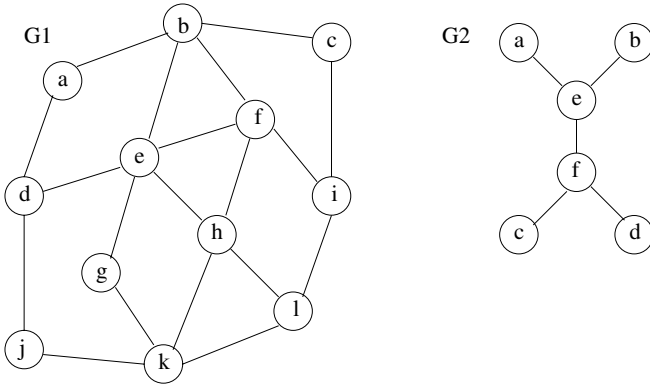


Fig. 1. Graphs for examples

For example, in graph $G1$ (Fig. 1) taking $S = \{a, c, h\}$, $R = \{b, e, k, i\}$ is a proper minimal separator of S , $S_b^R = \{a, c, h\}$, $S_e^R = \{a, h\}$, $S_j^R = \{a\}$ and $S_g^R = \emptyset$. The following theorem suggests that the concept of separator is fundamental to understand the facial structure of P_1 . The results of this subsection are proved in the appendix.

Theorem 2. *Every facet of $P_1(G)$ can be defined by a trivial inequality ($-x_i \leq 0$ or $x_i \leq 1$) or by an inequality in format $\sum_{i \in S} \beta_i x_i - \sum_{i \in R} \alpha_i x_i \leq 1$, where β and α are vectors of positive numbers and R is a proper separator of S .*

We define a family of potentially facet-defining inequalities.

Definition 4. *Let S be a vertex-set, where $|S| = p \geq 2$, and $R \in \Delta(S)$. The following inequalities are called p -separators, where α_i is an integer greater than $|S_i^R| - 1$.*

$$\sum_{i \in S} x_i - \sum_{i \in R} \alpha_i x_i \leq 1$$

Constraints F.2 are 2-separators. In graph $G1$, $x_a + x_f - x_b - x_e - x_j \leq 1$ is an example of such inequality. In the same graph, $x_a + x_f + x_g - x_b - 2x_e - 2x_k \leq 1$ is a 3-separator valid for $P_1(G1)$. Sometimes taking $\alpha_i = |S_i^R| - 1$ for all $i \in R$ does not yield a valid inequality for P_1 . For instance, in graph $G2$ (Fig. 1) taking $S = \{a, b, c, d\}$ and $R = \{e, f\}$, the 4-separator $\sum_{i \in S} x_i - x_e - x_f \leq 1$ is not valid for $P_1(G2)$. But 4-separators $\sum_{i \in S} x_i - 2x_e - x_f \leq 1$ and $\sum_{i \in S} x_i - x_e - 2x_f \leq 1$ are valid and facet-defining.

Theorem 3. *For each S and each $R \in \Delta(S)$, there is at least one p -separator inequality which is valid and facet-defining for $P_1(G)$.*

Corollary 1. *If the p -separator $\sum_{i \in S} x_i - \sum_{i \in R} (|S_i^R| - 1)x_i \leq 1$ is valid for $P_1(G)$, then it defines a facet of $P_1(G)$.*

The above theorem does not tell how to calculate the α coefficients of a p -separator facet. But given sets S and $R \in \Delta(S)$, the coefficients of a p -separator valid for P_1 can be computed by the following procedure:

Procedure Compute α

```

 $M \leftarrow \emptyset;$ 
For each  $i \in R$  {
    If (exists  $j \in M$  such that  $S_i^R \cap S_j^R = \emptyset$ )
        Then {  $\alpha_i \leftarrow |S_i^R|;$  }
        Else {  $\alpha_i \leftarrow |S_i^R| - 1;$   $M \leftarrow M \cup \{i\};$  }
}
    
```

As example, take graph G2 with $S = \{a, b, c, d\}$ and $R = \{e, f\}$. If we pick vertices in order (e, f) , the procedure yields $\alpha_e = 1$ and $\alpha_f = 2$. If the order is reversed, it yields $\alpha_f = 1$ and $\alpha_e = 2$. Note that if $p \leq 3$, the procedure always yields $\alpha_i = |S_i^R| - 1$ for all $i \in R$. Then, by Corollary 1, all such 2-separators and 3-separators define facets of $P_1(G)$. The above procedure is not optimal, if $p \geq 4$ the resulting p -separators are not always facet-defining.

3.2 From P_1 to P : p -Separators, Lifted p -Separators, and p -Crosses

If a certain p -separator defines a facet of $P_1(G)$, it may also define a facet of $P(G)$, so it is interesting to investigate when it happens. Also interesting is to investigate when this does not happen: in this case there is room for lifting the p -separator, producing a new joint inequality.

Definition 5. *The inequalities valid for $P(G)$ that can be obtained by lifting α coefficients of a p -separator facet of $P_1(G)$ or $P_2(G)$ are called lifted p -separators.*

The following definitions and lemma (proved in the appendix) are necessary for announcing and proving the main results of this section.

Definition 6. *Let T be a double v -tree of G and t a vertex in T . A sequence of nested double v -trees from t to T is any sequence of double v -trees $T_1, \dots, T_{|T|}$ such that $\{t\} = T_1 \subset \dots \subset T_{|T|} = T$. (In other words, for each i , $1 < i \leq |T|$, T_{i-1} may be obtained from T_i by removing a vertex distinct from t).*

Lemma 1. *Suppose G is biconnected. Given a double v -tree T of G and $t \in T$, it is always possible to construct a sequence of nested double v -trees from t to T .*

Definition 7. *A separator R of $S = \{s_1, \dots, s_p\}$ induces the following structure in G . Graph $G[V \setminus R]$ is split in a number of connected components: those containing exactly one vertex in S are called A-components and those with no vertices in S are called B-components. Define A_i as the vertex-set of the A-component containing s_i (when R is not proper some A-components do not exist). The vertex-sets corresponding to B-components are numbered arbitrarily as $B_1, \dots, B_q$. Note that $|S_i^R| = 0$ if i belongs to a B-component and $|S_i^R| = 1$ if i belongs to a A-component.*

Theorem 4. *Suppose G is biconnected. Let $\sum_{i \in S} x_i - \sum_{i \in R} (|S_i^R| - 1)x_i \leq 1$ be a valid p -separator, where R induces a structure without B -components and that every A -component is a double v -tree. If for every $i \in R$, there is a double v -tree U_i containing $\{i\} \cup S_i^R$ such that $U_i \cap R = \{i\}$, this inequality defines a facet of $P(G)$.*

Proof. Let $a^t x \leq a_0$ be a valid inequality for $P(G)$ such that

$$F = \{x \in P(G) \mid \sum_{i \in S} x_i - \sum_{i \in R} (|S_i^R| - 1)x_i = 1\} \subseteq F_a = \{x \in P(G) \mid a^t x = a_0\}.$$

We show that $a^t x \leq a_0$ is a scalar multiple of this p -separator. Consider the structure induced by R . For every A -component A_j , applying Lemma 1, construct a sequence of nested double v -trees from s_j to A_j . For each double v -tree T in this sequence, $\chi^T \in F$. So $a_{s_j} = a_0$ and for every $i \neq s_j \in A_j$, $a_i = 0$. For every vertex $i \in R$, $\chi^{U_i} \in F$, where U_i is the double v -tree given by hypothesis. So $a_i = -(|S_i^R| - 1)a_0$. $\square$

For example, in graph G_1 , the 2-separator $x_a + x_f - x_b - x_e - x_j \leq 1$, 3-separator $x_a + x_f + x_g - x_b - 2x_e - 2x_k \leq 1$ and 4-separator $x_a + x_c + x_g + x_h - 2x_b - 2x_e - x_i - 2x_k \leq 1$ define facets of $P(G_1)$. The converse of Theorem 4 does not hold even if $p = 3$, so we need some additional conditions to be able to lift α coefficients.

Theorem 5. *Suppose G is biconnected. Let $\sum_{i \in S} x_i - \sum_{i \in R} (|S_i^R| - 1)x_i \leq 1$ be a valid p -separator, where R induces a structure without B -components and that every A -component is a double v -tree. Suppose that there is a vertex $i \in R$ not satisfying Theorem 4 conditions. If $|S_i^R| = p$, the coefficient α_i can be lifted in at least one unit.*

Proof. We show that for every double v -tree T containing i , $\gamma_T = |S \cap T| - \sum_{i \in R \cap T} \alpha_i \leq 0$. If T contains S_i^R , $|S \cap T| = |S_i^R|$. As T must contain another vertex j of R , $\gamma_T \leq |S_i^R| - (|S_i^R| - 1) - (|S_j^R| - 1) = 2 - |S_j^R| \leq 0$. If T does not contain S_i^R , $\gamma_T \leq (|S_i^R| - 1) - (|S_j^R| - 1) \leq 0$. $\square$

If $p = 2$, the above results can be collected as follows:

Corollary 2. *Suppose G is biconnected. Let $\sum_{i \in S} x_i - \sum_{i \in R} x_i \leq 1$ be a 2-separator, where R induces a structure without B -components. This inequality defines a facet of $P(G)$ iff for every $i \in R$, there is a double v -tree U_i containing $\{i, s_1, s_2\}$ such that $U_i \cap R = \{i\}$*

Proof. As R is minimal, every vertex in R is adjacent to both A_1 and A_2 . So A_1 and A_2 are double v -trees. So the “if” part follows from Theorem 4. For each $i \in R$, $|S_i^R| = 2$. So the “only if” part follows from Theorem 5. $\square$

For examples, in graph G_1 , the 2-separator $x_a + x_c - x_e - x_f - x_h - x_l \leq 1$ do not define a facet of $P(G_1)$, by Corollary 2, α_e or α_h can be lifted to 0. The

3-separator $x_a + x_c + x_g - x_b - 2x_e - 2x_k \leq 1$ also do not define a facet $P(G1)$, by Theorem 5, α_e can be lifted to 1.

In graph G3 (Fig. 2), the 2-separator $x_a + x_b - x_d - x_e - x_f \leq 1$ is not facet-defining, although it satisfies the conditions of Corollary 2, except that R induces a structure with B-components. This 2-separator is dominated by the lifted 3-separator $x_a + x_b + x_c - x_d - x_e - x_f \leq 1$ (from 3-separator $x_a + x_b + x_c - 2x_d - 2x_e - 2x_f \leq 1$). In graph G4 (Fig. 2), 3-separator $x_a + x_c + x_e - x_b - x_d - x_f - x_h \leq 1$ is not facet-defining, although it satisfies the conditions of Theorem 4, except that the A-component formed by $\{c, g\}$ is not a double v-tree. This 3-separator is dominated by $x_a + x_c + x_g + x_e - x_b - x_d - x_f - x_h \leq 1$.

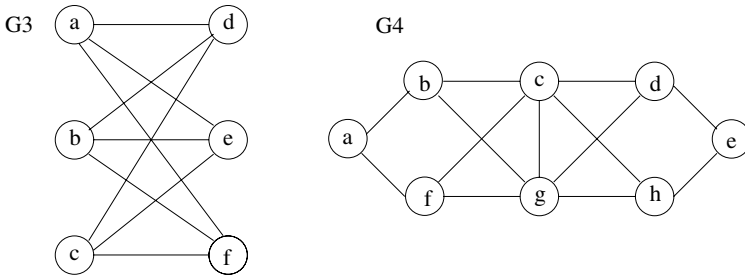


Fig. 2. Graphs for examples

The concept of lifted p -separators acts as a convenient classifying device, permitting us to view lots of joint inequalities in a single framework. Even though Theorem 5 does not cover all cases when a p -separator facet of $P_1(G)$ can be lifted. It is certainly possible to improve these results, but difficulties appear. For instance, when $p \geq 4$ it is not clear even how to calculate the original α coefficients of a p -separator facet of $P_1(G)$. Also, it is usual that several such coefficients can be lifted at once. Conditions for this “multilifting” are more complicated. Most of all, it is cumbersome to design separation procedures for lifted p -separators departing from the original p -separator.

We give a direct characterization of an important subfamily of lifted p -separators that permitted us to overcome such difficulties. These inequalities turned out to be essential to good branch-and-cut performance.

Definition 8. Let $S = \{s_1, \dots, s_p\}$ and $R = \{r_1, \dots, r_p\}$ be two disjoint vertex-sets with $p \geq 2$. Suppose that for any integers a, b, c, d in the range $\{1, \dots, p\}$, with $a \leq b < c \leq d$, there are no two vertex-disjoint paths joining s_a to s_c and r_b to r_d (in other words, the vertices in every path from r_b to r_d define a separator of $\{s_a, s_c\}$ and vice-versa). Then the following inequality is called a p -cross

$$\sum_{i \in S} x_i - \sum_{i \in R} x_i \leq 1$$

In graph $G1$, $x_e + x_j - x_d - x_g \leq 1$ is a 2-cross and $x_a + x_c + x_l + x_j - x_b - x_i - x_k - x_d \leq 1$ is a 4-cross. Most p -crosses are lifted p -separators, it suffices to “complete” R until a proper minimal separator of S is found. For instance, 3-cross $x_a + x_c + x_k - x_b - x_l - x_d \leq 1$ is a lifting of 3-separator $x_a + x_c + x_k - 2x_b - x_f - x_l - x_d \leq 1$ facet of $P_1(G1)$ or $-x_b - x_l - x_d + x_a + x_c + 2x_e + x_f + x_k \leq 4$ facet of $P_2(G1)$.

Theorem 6. *The p -crosses are valid for P .*

Proof. By induction on p . The 2-crosses are clearly valid. Suppose that $\sum_{i=1}^p x_{s_i} - \sum_{i=1}^p x_{r_i} \leq 1$ is a p -cross with $p \geq 3$. By induction hypothesis, $(p-1)$ -crosses $\sum_{i=1}^{p-1} x_{s_i} - \sum_{i=1}^{p-1} x_{r_i} \leq 1$ and $\sum_{i=2}^p x_{s_i} - \sum_{i=2}^p x_{r_i} \leq 1$ are valid. Summing these inequalities with 2-cross $x_{s_1} + x_{s_p} - x_{r_1} - x_{r_p} \leq 1$, dividing the result by 2 and rounding down the right-hand-side, the desired p -cross is obtained. $\square$

Theorem 7. *If G is biconnected, every 2-cross defines a facet of $P(G)$.*

Proof. Let $x_{s_1} + x_{s_2} - x_{r_1} - x_{r_2} \leq 1$ be a 2-cross and $a^t x \leq a_0$ be a valid inequality for $P(G)$ such that

$$F = \{x \in P(G) \mid x_{s_1} + x_{s_2} - x_{r_1} - x_{r_2} = 1\} \subseteq F_a = \{x \in P(G) \mid a^t x = a_0\}.$$

We show that $a^t x \leq a_0$ is an scalar multiple of this 2-cross. As G is biconnected, there are two vertex-disjoint paths, except in the endpoints, joining r_1 to r_2 . Let I_1 and I_2 be the set of vertices of any two such paths. As I_1 and I_2 are separators of S , it is not possible to have $S \subseteq I_1$ or $S \subseteq I_2$.

Case (a): I_1 or I_2 is a proper separator of S .

Suppose I_1 is a proper separator of S and consider the induced structure. As G is biconnected, each component is adjacent to at least two vertices in I_1 . We assert that at least one of the two A-components is adjacent to both r_1 and r_2 . If this is not true, the path corresponding to I_2 must pass through a B-component and I_1 would not be a separator of S . Suppose A_2 is an A-component adjacent to r_1 and r_2 .

Subcase (a.1): A_1 is adjacent to a vertex in $I'_1 = I_1 \setminus \{r_1, r_2\}$. In this case, $C_1 = A_1 \cup I'_1 \cup \{B_k \mid B_k \text{ is adjacent to } I'_1\}$ and $C_2 = A_2$ are double v-trees. Construct a sequence of nested double v-trees from s_1 to C_1 and from s_2 to C_2 . For each T in these sequences, $\chi^T \in F$. So $a_{s_1} = a_{s_2} = a_0$ and $a_i = 0$ if $i \in (C_1 \cup C_2) \setminus S$. As $\chi^{C_1 \cup C_2 \cup \{r_1\}} \in F$ and $\chi^{C_1 \cup C_2 \cup \{r_2\}} \in F$, $a_{r_1} = a_{r_2} = -a_0$. Each component B_k not adjacent to I'_1 must be adjacent to both r_1 and r_2 . Let j be a vertex in B_k adjacent to r_1 . Construct a sequence of nested double v-trees from j to B_k . For each T in this sequence $U_T = C_1 \cup C_2 \cup \{r_1\} \cup T$ is also a double v-tree and $\chi^{U_T} \in F$. So $a_i = 0$, for all $i \in B_k$.

Subcase (a.2): A_1 is not adjacent to I'_1 . In this case, A_1 is adjacent to r_1 and r_2 . Let $C_1 = A_1$, $C_2 = A_2$ and proceed similar to case a.1, concluding that $a_{s_1} = a_{s_2} = a_0$, $a_i = 0$ if $i \in (C_1 \cup C_2) \setminus S$ and $a_{r_1} = a_{r_2} = -a_0$. Now define C_3 as $I'_1 \cup \{B_k \mid B_k \text{ is adjacent to } I'_1\}$. Let j be a vertex in I'_1 adjacent to r_1 . Construct

a sequence of nested double v-trees from j to C_3 . For each T in this sequence $U_T = C_1 \cup C_2 \cup \{r_1\} \cup T$ is also a double v-tree and $\chi^{U_T} \in F$. So $a_i = 0$, for each $i \in C_3$. B-components adjacent to both r_1 and r_2 are treated in a similar way.

Case (b): I_1 and I_2 are not proper separators of S .

Suppose I_1 contains s_1 and I_2 contains s_2 . Consider the structure induced by separator I_1 . As I_2 contains s_2 , A_2 is adjacent to r_1 and r_2 . Let $C_1 = I_1' \cup \{B_k \mid B_k \text{ is adjacent to } I_1'\}$, $C_2 = A_2$ and proceed similar to subcase a.1. B-components adjacent to both r_1 and r_2 are also treated like in a.1. $\square$

Using similar techniques it is possible to prove that under mild conditions p -crosses with $p \geq 3$ also define facets of $P(G)$. For instance, supposing G is biconnected and planar, all p -crosses are facet-defining.

4 Separation Procedures

The 2-separators can be separated by applying the min-cut algorithm over a suitable network. When $p \geq 3$, the procedures to find violated p -separators inequalities are harder to devise. One heuristic procedure to find such a p -separator is by the successive application of the 2-separator procedure. Note that if $p \geq 4$ there is the additional complication of calculating proper α coefficients.

The p -crosses can be separated in polynomial time over any graph G , for fixed p . This follows from definition 8, since there exists an $O(|V| \cdot |E|)$ algorithm to determine if it is possible to connect two given pairs of vertices by vertex-disjoint paths (Shiloach [7]). However, far better p -cross separation schemes may exist if the particular topological structure of G is considered. If G is planar and supposing that no 2-separator is violated, a violated p -cross only occurs when the sequence $s_1, r_1, \dots, s_p, r_p$ happen in this cyclic order on some face f of G . By this property, p -cross inequalities can be separated in $O(|V|)$ over planar graphs, by dynamic programming.

5 Computational Results

The branch-and-cut for 2VPST was implemented in C++ using the ABACUS framework [8] and CPLEX 3.0 LP solver, running on a Sun ULTRA workstation. The code used 2-separators, 3-separators (heuristically separated) and p -crosses.

In all experiments described in table 1, G is a square grid graph (grid graphs are typical in circuit layout applications). Two kind of instances were generated:

- Unweighted instances with $|N_1| = |N_2| = |V|/4$. The terminals in each of these sets are randomly chosen among the vertices of G . Vertices in N_1 have weight -1 and vertices in N_2 have weight 1. These instances are referenced as `u_n_Inst`, where $n = |V|$ and `Inst` is the instance number. The value z^* gives the number of unconnected terminals in an optimal solution.

- Weighted instances. For each unweighted instance we generated a corresponding weighted instance, by picking an integer w_i from an uniform distribution in the range $[-10, -1]$ if $i \in N_1$ and from $[1, 10]$ if $i \in N_2$. The value z^* gives the sum of the absolute weights of the unconnected terminals in an optimal solution. Weighted instances are referenced as *w_n_Inst*.

Table 1. Branch-and-Cut Results

Instance	LB _r	z^*	Nd LPs			T	Instance	LB _r	z^*	Nd LPs			T
u1600_1	45.7	46	4	96	20		w1600_1	222.0	222	2	35	9	
u1600_2	40.0	40	2	40	8		w1600_2	218.0	218	1	25	5	
u2500_1	46.0	46	2	56	33		w2500_1	233.0	233	2	37	14	
u2500_2	55.0	55	3	49	32		w2500_2	227.5	228	3	39	17	
u3600_1	76.5	77	2	45	41		w3600_1	364.0	364	2	48	33	
u3600_2	74.0	74	4	57	50		w3600_2	311.0	311	2	74	54	
u4900_1	99.1	100	4	116	170		w4900_1	420.0	420	2	77	130	
u4900_2	92.0	92	3	66	108		w4900_2	400.0	400	3	53	90	
u6400_1	104.0	104	3	126	443		w6400_1	436.0	436	3	106	295	
u6400_2	127.0	128	4	123	322		w6400_2	534.0	534	3	128	367	
u8100_1	131.5	132	4	217	1207		w8100_1	549.5	550	4	113	563	
u8100_2	115.0	115	6	125	713		w8100_2	467.0	467	3	82	264	
u10000_1	159.8	161	8	120	676		w10000_1	685.0	685	3	60	345	
u10000_2	151.5	152	4	157	991		w10000_2	632.5	634	7	116	621	

In Table 1, the **LB_r** column presents the lower bound obtained in the root node, **z^*** is the value of the optimal integer solution, **Nd** is the number of nodes explored in the branch-and-bound tree, **LPs** is the total number of solved LPs and **T** is the total cpu time in seconds spent by the algorithm. Remark that **LB_r** bounds includes *p*-crosses and 3-separators. The lower bounds given by the linear relaxation of formulation F alone are rather weak, for example, in instance u1600_1 it is 8.7. Figure 3 shows an optimal solution of instance u1600_1, the black boxes represent the terminals in N_1 , the empty boxes are the terminals in N_2 , the full lines are edges of a spanning tree for $G[T_1]$ and the dotted lines a spanning tree for $G[T_2]$.

The code turned out to be robust under changes on terminal placement, terminal density and weight distribution. In fact, we used the same set of parameters for solving all instances in table 1, weighted and unweighted. An interesting experience is to generate a random terminal placement avoiding the most obvious causes of infeasibility: terminals on the outer face, two pairs of opposite terminals crossed over the same grid square and terminals blocked by its neighbors. An optimal solution of such an instance (called t1600_1) with 800 terminals and unitary weights is shown in Fig. 4. The unconnected terminals are in coordinates (4,33), (8,21), (33,36) and (34,15). This solution was obtained in 21 seconds.

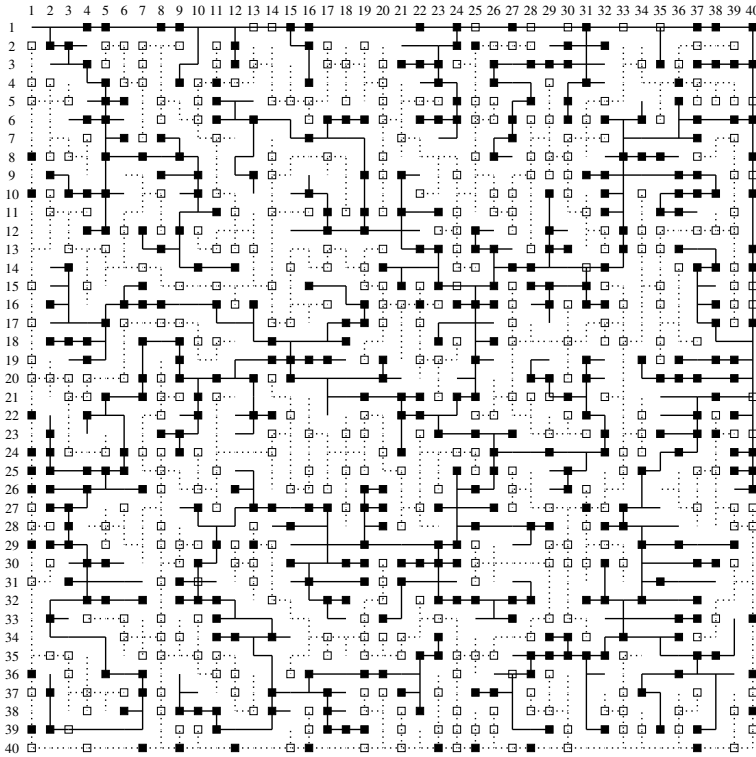


Fig. 3. An optimal solution of instance u1600_1, $z^* = 46$.

Appendix

Proof (Theorem 2). Suppose that a facet F of $P_1(G)$ is defined by $a^t x \leq a_0$. The right-hand-side a_0 must be non-negative, otherwise this inequality would not be valid for the empty v-tree. So, w.l.o.g. we can suppose that $a_0 = 0$ or $a_0 = 1$. Let S be the vertex-set corresponding to positive coefficients in a and R the vertex-set corresponding to negative coefficients.

Case (a): $a_0 = 0$. The set S must be empty, otherwise $a^t x \leq 0$ would not be valid for the v-tree formed by a single vertex in S . The set R must be a singleton $\{u\}$, since if there are other vertices in R , $a^t x \leq 0$ would be dominated by $-x_u \leq 0$. So F can be defined by this trivial inequality.

Case (b): $a_0 = 1$. S can not be empty. If S is a singleton $\{u\}$, F can be defined by the trivial inequality $x_u \leq 1$. Suppose $|S| \geq 2$. Graph $G[V \setminus R]$ is split into a number of connected components. No two vertices in S can belong to the same such component, since if u and w are two vertices in S in the same component then $a^t x \leq 1$ would be a linear combination of valid inequalities $\sum_{i \in S \setminus \{u, w\}} a_i x_i + (a_u + a_w)x_u + \sum_{i \in R} a_i x_i \leq 1$ and $\sum_{i \in S \setminus \{u, w\}} a_i x_i + (a_u + a_w)x_w + \sum_{i \in R} a_i x_i \leq 1$. Therefore R is a proper separator of S . $\square$

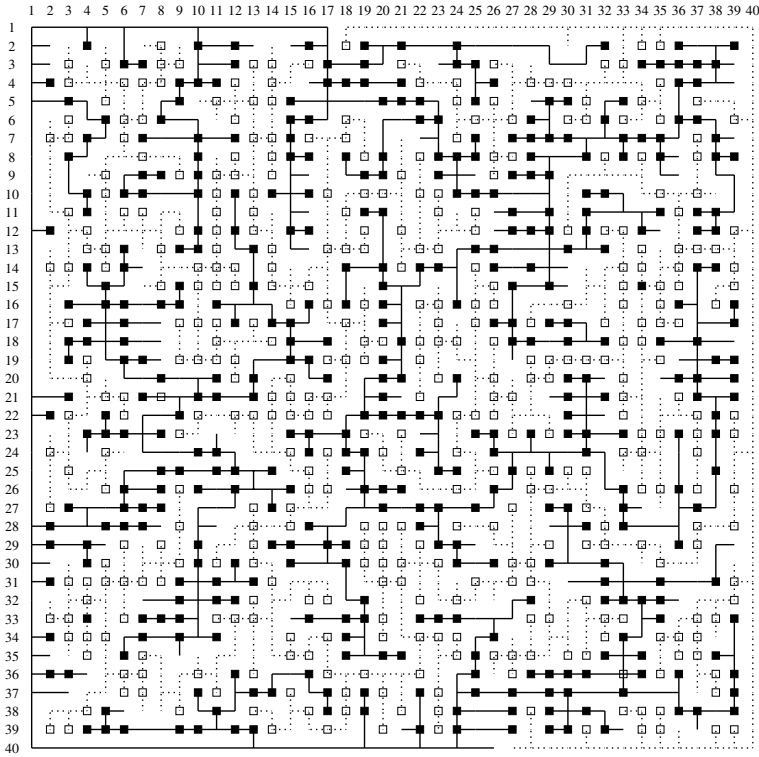


Fig. 4. An optimal solution of instance t1600_1, $z^* = 4$.

The following lemma and definition are used in the proof of Theorem 3.

Lemma 2. (*Sequential Maximum Lifting Lemma, pages 261-263 of [5]*) Let P be a polyhedron in $\mathbb{R}^n$. Let N be the set of indices $\{1, \dots, n\}$ and Z_0 a subset of N . Define P^{Z_0} as the polyhedron $P \cap \{x \in \mathbb{R}^n | x_i = 0, i \in Z_0\}$. Suppose $\sum_{i \in N \setminus Z_0} a_i x_i \leq a_0$ is a valid inequality for P^{Z_0} defining a face of dimension d . Let $\sum_{i \in N \setminus Z_0} a_i x_i + \sum_{i \in Z_0} m_i x_i \leq a_0$ be any inequality obtained by a sequential maximum lifting of one coefficient in Z_0 at a time (in any order). This lifted inequality defines a face of P of dimension $d + |Z_0|$, and if the original inequality defines a facet of P^{Z_0} , the lifted inequality defines a facet of P .

Definition 9. Let T be a v -tree of G and t a vertex in T . A sequence of nested v -trees from t to T is any sequence of v -trees $T_1, \dots, T_{|T|}$ such that $\{t\} = T_1 \subset \dots \subset T_{|T|} = T$. Note that given v -tree T and t , is always possible to construct such a sequence.

Proof (Theorem 3). Consider the structure induced by R (see definition 7). Inequality $\sum_{i \in S} x_i \leq 1$ is valid for $P_1(G)^{Z_0}$, where $Z_0 = R \cup (\cup_{j=1}^q B_j)$. Define

$F = \{x \in P_1(G)^{Z_0} \mid \sum_{i \in S} x_i = 1\}$. For each $j \in \{1, \dots, p\}$, A_j is a v-tree. Construct a sequence of nested v-trees from s_j to A_j . For every T in these sequences, $\chi^T \in F$. So there are $\sum_{j=1}^p |A_j|$ affine independent vectors in F and $\sum_{i \in S} x_i \leq 1$ defines a facet of $P_1(G)^{Z_0}$.

Number the vertices of R from r_1 to $r_{|R|}$. Define $R_0 = \emptyset$ and R_j as the set $\{r_1, \dots, r_j\}$. For each $k \in \{1, \dots, q\}$, define $f(k)$ as the smaller number of a vertex in R adjacent to B-component B_k . Define $W_j = R_j \cup \{B_k \mid f(k) \leq j\}$ and $Z_j = Z_0 \setminus W_j$. We show that for each $j \in \{0, \dots, |R|\}$, there is an inequality $\sum_{i \in S} x_i - \sum_{i \in R_j} \alpha_i x_i \leq 1$, where each α_i is integer and satisfies $\alpha_i \geq |S_i^R| - 1$, defining a facet of $P_1(G)^{Z_j}$.

This is true when $j = 0$. Suppose $1 \leq j \leq |R|$. Pick the inequality corresponding to $j - 1$. First lift the coefficient of r_j . As there exists a v-tree T_j such that $T_j \cap R = \{r_j\}$, $T_j \cap S = S_j^R$ and $T_j \cap (\cup_{j=1}^q B_j) = \emptyset$, $\alpha_j \geq |S_j^R| - 1$. As the lifting is maximum, α_j is integral and there exists a v-tree $U \subseteq (\cup_{j=1}^p A_j \cup W_{j-1} \cup \{r_j\})$ such that $r_j \in U$ and $|S \cap U| - \sum_{i \in R_j \cap U} \alpha_i = 1$. For each B-component B_k such that $f(k) = j$, take a vertex $t \in B_k$ adjacent to r_j . Construct a sequence of nested v-trees from t to B_k . Make a maximum lifting of the coefficients of vertices in B_k in the same order this vertices are introduced in the sequence. Since $U \cup T$ is a v-tree for every T in the sequence, all such coefficients must be zero. Repeating this procedure until $j = |R|$, a p -separator defining a facet of $P_1(G)$ is obtained. $\square$

Proof (Correctness of procedure Calculate α). Let U be a v-tree and let $Q = R \cap U$. We must show that $|S \cap U| - \sum_{i \in Q} \alpha_i \leq 1$. Assume that Q is not empty, otherwise this inequality is clearly true. Sort the vertices in Q as $q_1, \dots, q_{|Q|}$, using the same order in which these vertices were considered in procedure Calculate α . Define Q_i as the set $\{q_1, \dots, q_i\}$ and $S_{Q_i}^R$ as $\cup_{j \in Q_i} S_j^R$.

We assert that for each $i \in \{1, \dots, |Q|\}$, $\gamma_i = |S_{Q_i}^R \cap U| - \sum_{i \in Q_i} \alpha_i \leq 1$. This is true when $i = 1$. Suppose $1 < i \leq |Q|$. If $\alpha_i = |S_i^R|$, then $\gamma_i \leq \gamma_{i-1} \leq 1$. Suppose $\alpha_i = |S_i^R| - 1$. The procedure assures that if there exists a vertex $j \in Q_{i-1}$ with $\alpha_j = |S_j^R| - 1$ then $S_i^R \cap S_j^R \neq \emptyset$, and $\gamma_i \leq \gamma_{i-1} \leq 1$. If there is no such vertex $\gamma_{i-1} \leq 0$, so $\gamma_i \leq 1$. As $|S \cap U| - \sum_{i \in Q} \alpha_i \leq \gamma_{|Q|} \leq 1$, the result follows. $\square$

Proof (Lemma 1). If $T = \{t\}$ the result is immediate. Otherwise, suppose that a partial sequence $T = T_{|T|} \supset \dots \supset T_i$, $|T| \geq i > 1$, is already formed. Vertices in double v-tree T_i , different from t , but adjacent to a vertex in $V \setminus T_i$ will be called *candidate vertices* (if $T = V$, all vertices in $V - t$ are candidates). As G is biconnected, there is at least one candidate vertex. We show that is always possible to choose a candidate to be removed from T_i to form double v-tree T_{i-1} .

Let v_0 be a candidate. If $T_i - v_0$ is a v-tree, v_0 is chosen. Otherwise $G[T_i - v_0]$ is separated into $j \geq 2$ connected components with vertex-sets $C_1^1, \dots, C_j^1$. As T_i is a v-tree, at least one vertex in C_j^1 is adjacent to v_0 . As G is biconnected, at least one vertex in each C_j^1 is adjacent to $V \setminus T_i$. Adjust notation for C_1^1 to be a set not containing t . Let v_1 be a candidate vertex in C_1^1 . $G[C_1^1 - v_1]$ is separated

into $j \geq 0$ connected components with vertex-sets $C_1^2, \dots, C_j^2$. As C_1^1 is a v-tree, at least one vertex in each C_j^2 is adjacent to v_1 . As G is biconnected, at least one vertex in each C_j^2 is adjacent to v_0 or to $V \setminus T_i$. If at least one vertex in each C_j^2 is adjacent to v_0 (or $j = 0$), v_1 is chosen. Otherwise, adjust notation for C_1^2 to be a set containing a candidate vertex.

By repeating this procedure, we get a sequence $v_1, \dots, v_k$ of candidate vertices contained in smaller sets C_1^k . If each component of $G[C_1^k - v_k]$ is adjacent to $T_i \setminus C_1^k$, v_k is the chosen candidate and $T_i - v_k$ is a v-tree. In the worst case, the procedure is repeated until $C_1^k = \{v_k\}$. $\square$

Acknowledgments: The first author was supported by CNPq Bolsa de Doutorado. The second author was partially supported by CNPq Bolsa de Produtividade em Pesquisa 300475/93-4.

References

1. M. Grötschel, A. Martin and R. Weismantel, *Packing Steiner Trees: polyhedral investigations*, Mathematical Programming, Vol. 72, 101-123, 1996.
2. M. Grötschel, A. Martin and R. Weismantel, *Packing Steiner Trees: a cutting plane algorithm and computational results*, Mathematical Programming, Vol. 72, 125-145, 1996.
3. B. Korte, H. Prömel and A. Steger, *Steiner Trees in VLSI-Layout*, in B. Korte, L. Lovász, H. Prömel, A. Schrijver (Eds.) "Paths, Flows and VLSI-Layout", Springer-Verlag, Berlin, 1989.
4. T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley, Chichester, 1990.
5. G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York, 1988.
6. M. Poggi de Aragão and E. Uchoa, The γ -Connected Assignment Problem, To appear in European Journal of Operational Research, 1998.
7. Y. Shiloach, *A Polynomial Solution to the Undirected Two Paths Problem*, Journal of the ACM, Vol. 27, No.3, 445-456, 1980.
8. S. Thienel, *ABACUS - A Branch-And-Cut System*, PhD thesis, Universität zu Köln, 1995.

Author Index

- Aardal, K. 1
Ageev, A.A. 17
Ahuja, R.K. 31
Amaldi, E. 45
Atamtürk, A. 60
Bixby, R.E. 1
Cai, M. 73
Caprara, A. 87
Chudak, F.A. 99
Cunningham, W.H. 114
Cvetković, D. 126
Čangalović, M. 126
Deng, X. 73
Eisenbrand, F. 137
Fischetti, M. 87
Fleischer, L. 151
Fonlupt, J. 166
Frank, A. 183, 191
Halperin, E. 202
Hartmann, M. 218
Hartvigsen, D. 234
Helmberg, C. 242
Hochbaum, D.S. 31
Hurkens, C.A.J. 1
Ibaraki, T. 377
Iwata, S. 259
Jordán, T. 183, 273
Kashiwabara, K. 289
Király, Z. 191
Klau, G.W. 304
Klein, P. 320
Kolliopoulos, S.G. 328
Kovačević-Vujčić, V. 126
Lenstra, A.K. 1
Letchford, A.N. 87
McCormick, S.T. 259
Mahjoub, A.R. 166
Melkonian, V. 345
Mutzel, P. 304, 361
Nagamochi, H. 377
Nakamura, M. 289
Nemhauser, G.L. 60
Noga, J. 391
Orlin, J.B. 31
Pfetsch, M.E. 45
Poggi de Aragão, M. 439
Queyranne, M. 218
Savelsbergh, M.W.P. 60
Schulz, A.S. 137
Sebő, A. 400
Seiden, S. 391
Sethuraman, J. 429
Shigeno, M. 259
Smeltink, J.W. 1
Stein, C. 328
Sviridenko, M.I. 17
Szigeti, Z. 183, 415
Takabatake, T. 289
Tan, W.-P. 429
Tang, L. 114
Tardos, É. 345
Teo, C.-P. 429
Trotter, L.E., Jr. 45
Uchoa, E. 439
Wang, Y. 218
Weiskircher, R. 361
Williamson, D.P. 99
Young, N. 320
Zang, W. 73
Zwick, U. 202