

# Lecture Notes in Artificial Intelligence

1898

Subseries of Lecture Notes in Computer Science

Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science

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

**Springer**

*Berlin*

*Heidelberg*

*New York*

*Barcelona*

*Hong Kong*

*London*

*Milan*

*Paris*

*Singapore*

*Tokyo*

Enrico Blanzieri   Luigi Portinale (Eds.)

# Advances in Case-Based Reasoning

5th European Workshop, EWCBR 2000  
Trento, Italy, September 6-9, 2000  
Proceedings



Springer

## Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA  
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

## Volume Editors

Enrico Blanzieri  
Istituto Trentino di Cultura  
Centre for Scientific and Technological Research (ITC-irst)  
via Sommarive 18, 38050 Povo (Trento), Italy  
E-mail: blanzieri@irst.itc.it

Luigi Portinale  
DISTA, University of Eastern Piedmont "Amedeo Avogadro"  
C.so Borsalino 54, 15100 Alessandria, Italy  
E-mail: portinal@mf.n.unipmn.it

Cataloging-in-Publication Data applied for

## Die Deutsche Bibliothek - CIP-Einheitsaufnahme

**Advances in case based reasoning : 5th European workshop ; proceedings /  
EWCBR 2000, Trento, Italy, September 6 - 9, 2000. Enrico Blanzieri ;  
Luigi Portinale (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ;  
Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000  
(Lecture notes in computer science ; Vol. 1898 : Lecture notes in  
artificial intelligence)  
ISBN 3-540-67933-2**

CR Subject Classification (1998): I.2, F.4.1

ISBN 3-540-67933-2 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 New York  
a member of BertelsmannSpringer Science+Business Media GmbH  
© Springer-Verlag Berlin Heidelberg 2000  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein  
Printed on acid-free paper      SPIN 10722565      06/3142      5 4 3 2 1 0

# Preface

The papers collected in this volume were presented at the fifth European Workshop on Case-Based Reasoning (EWCBR2K) held Trento, Italy. EWCBR2K was the last episode of a series of very successful meetings held previously in Kaiserslautern (1993), Paris (1994), Lausanne (1996) and Dublin (1998) that traditionally attract European and international researchers and practitioners in case-based reasoning.

Case-based reasoning is a recent field of research with a long history of applications. The organization of the volume is consistent with this vision of the field and, after an initial part devoted to the invited papers, the two main parts contain research and application papers respectively. The overall result is a representative snapshot of current research and state-of-the-art applications. The merit for this volume goes to the authors of the papers, the Program Committee and all the additional reviewers whose combined effort produced this high-quality scientific work.

The organizers would like to thank all the people who contributed to the success of the workshop: in particular, Francesco Ricci who originally proposed to have EWCBR2K in Italy; Fausto Giunchiglia, Paolo Traverso and Oliviero Stock who backed the proposal and assured the institutional commitment of ITC-irst and University of Trento; Paolo Avesani, Mehmet Goeker and Michel Manago who developed the traditional industrial-day into the first edition of Innovative Customer Centered Applications (ICCA), a co-located event with a focused target on industry people; Giampaolo Avancini, Sara Beatrici, Morena Carli, Carola Dori, Alessandro Tuccio, of ITC-IRST and Fabrizio Di Bartolo of University of the Eastern Piedmont who were effective, professional and friendly while providing technical and administrative support for the organization of the event.

We would like to offer thanks for the support provided by EWCBR-98, MLNET, the Italian Association for Artificial Intelligence (AI\*IA) and the Department of Advanced Sciences and Technologies (DISTA) of the University of Eastern Piedmont (Università del Piemonte Orientale "A. Avogadro"). A special thank to the invited speakers Barry Smyth and Qiang Yang and to all the chairs of the sessions and finally, to Springer-Verlag for their help and their enthusiastic agreement on the publication of this volume.

June 2000

Enrico Blanzieri and Luigi Portinale

## **Program Chairs**

Enrico Blanzieri, ITC-IRST, Italy

Luigi Portinale, Università del Piemonte Orientale “A. Avogadro”, Italy

## **Program Commitee**

Agnar Aamodt, Norwegian University of Science and Technology, Norway

Robert J. Aarts, Nokia Telecommunications, Finland

Klaus-Dieter Althoff, Fraunhofer IESE, Germany

David W. Aha, Naval Research Laboratory, USA

Kevin Ashley, University of Pittsburgh, USA

Paolo Avesani, ITC-irst, Italy

Ralph Bergmann, University of Kaiserslautern, Germany

Carlos Bento, University of Coimbra, Portugal

Karl L. Branting, University of Wyoming, USA

Susan Craw, Robert Gordon University, Scotland UK

Padraig Cunningham, University of Dublin, Ireland

Mehmet Goeker, DaimlerChrysler, USA

Boi Faltings, EPF Lausanne, Switzerland

Mark Keane, University College Dublin, Ireland

David Leake, Indiana University, USA

Brian Lees, University of Paisley, UK

Ramon Lopez de Mantaras, IIIA-CSIC, Catalonia, Spain

Michel Manago, Acknosoft, France

Alain Mille, Université Claude Bernard Lyon 1, France

Bart Netten, TNO-TPD Delft, The Netherlands

Enric Plaza, IIIA-CSIC, Catalonia, Spain

Francesco Ricci, ITC-irst, Italy

Jerzy Surma, Technical University of Wroclaw, Poland

Henry Tirri, University of Helsinki, Finland

Brigitte Trousse, INRIA Sophia Antipolis, France

Maarten van Someren, University of Amsterdam, The Netherlands

Ian Watson, University of Auckland, New Zealand

Stefan Wess, TecInno, Germany

## **Additional Reviewers**

Josep Ll. Arcos, Stefano Aguzzoli, Ole André Brevik, Riccardo Bellazzi, Paolo Busetta, Amilcar Cardoso, Alessandro Ebranati, Pere Garcia, Paulo Gomes, Christiane Gresse von Wangenheim, Conor Hayes, Penousal Machado, Paolo Massa, Stefania Montani, Petri Myllymaki, Markus Nick, Pavel Petrovic, Pinar Ozturk, Anna Perini, Thomas Roth-Berghofer, Rainer Schmidt, Sascha Schmitt, Andrea Sboner, Barry Smyth, Frode Sørmo, Francisco Camara Pereira, Ivo Vollrath, David C. Wilson, Martin Würthner

# Competence Models and Their Applications

Barry Smyth

Smart Media Institute, Department of Computer Science, University College Dublin  
Belfield, Dublin 4, Ireland

[Barry.Smyth@ucd.ie](mailto:Barry.Smyth@ucd.ie)

<http://www.cs.ucd.ie/staff/bsmyth>

For the artificial intelligence community the competence or coverage of a given problem solving system is a fundamental evaluation criterion. This is of course also true for case-based reasoning (CBR) systems, which solve new problems by reusing and adapting the solutions of similar problems stored as cases in a case-base [3,4,12]. However, our current understanding of the competence issue is limited. Usually the competence of a system is demonstrated empirically, for example, by measuring the percentage problem solving success of a system. Unfortunately, this type of approach reveals little more than the tip of the iceberg when it comes to the true nature of competence in CBR. For example, it reveals nothing about the underlying source of competence for a given system, and carries only limited predictive power. This makes it impossible to determine how the competence of a given system will vary under different run-time conditions (e.g., changes in the case-base, adaptation knowledge, or target problem space).

Recently the issue of competence has received further attention from the perspective of the so-called case-base maintenance problem, that is, the issue of how best to manage the organisation and contents of a case-base in order to optimise future reasoning performance (eg., [5,10,16]). In practice, this has involved the development of a range of policies for controlling the growth of case-bases and the organisation and indexing of the knowledge stored in case-bases. For example, a variety of case addition and deletion policies have been proposed to manage case-base growth (see [11,14,16]) alongside policies for revising (and deriving) case indices (see [1,2,9]).

The importance of case competence has been brought sharply into focus since many maintenance policies are directly linked to heuristics that attempt to measure case competence to guide the maintenance procedure (eg., [11,16]). However, these competence heuristics have typically provided only coarse-grained estimates of competence; for example, Smyth and Keane [11] employed a category-based competence model that classifies cases as belonging to one of only four possible competence categories.

This talk will focus on our recent work on the development of explicit, algorithmic models of competence for case-based reasoning systems. We suggest that such models are fundamentally important, and ultimately essential, in developing generic solutions to the maintenance problem. For example, we will describe how our competence model can be used to provide innovative solutions to problems such as; the construction of compact competent case-bases [7,8,14],



the case retrieval problem [15], and the provision of case authoring support and case-base visualisation [6,13].

## References

1. Aha, D.W. & Breslow, L.A.: *Refining Conversational Case Libraries*, Proceedings of the International. Conference on Case-Based Reasoning, 1997, pp. 267–278. 1
2. Fox, S. & Leake, D.B.: *Using Introspective Reasoning to Refine Indexings*, Proceedings of the 14th International Joint Conference on Artificial Intelligence (Mellish, C. ed.), Morgan Kaufmann, 1995, pp. 391–397. 1
3. Kolodner, J.: *Case-Based Reasoning*, Morgan Kaufmann, 1993. 1
4. Leake, D.B: (ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, MIT Press, 1996. 1
5. Leake, D.B. & Wilson, D.C.: *Categorizing Case-Based Maintenance: Dimensions and Directions.*, Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence (Smyth, B. & Cunningham, P. eds.), Springer Verlag, 1998, pp. 196–207. 1
6. McKenna, E. and Smyth, B.: *An Interactive Visualisation Tool for Case-Based Reasoners*, Applied Intelligence: Special Issue on Interactive Case-Based Reasoning (2000). 2
7. McKenna, E. and Smyth, B.: *Competence-guided Editing Methods for Lazy Learning*, Proceedings of the 14th European Conference on Artificial Intelligence, (2000). 1
8. McKenna, E. and Smyth, B.: *Competence-guided Case-Base Editing Techniques*, Proceedings of the 5th European Workshop on Case-Based Reasoning, (2000). 1
9. Racine, K. & Yang, Q.: *Maintaining Unstructured Case Bases.*, Case-Based Reasoning Research and Development. (Leake, D.B. & Plaza, E.: eds.), Springer Verlag, 1997, pp. 553–564. 1
10. Smyth, B.: *Case-Based Maintenance*, Tasks and Methods in Applied Artificial Intelligence. Lecture Notes in Artificial Intelligence (del Pobil, A.P., Mira, J. & Ali, M. eds.), Springer Verlag, 1998, pp. 507–516. 1
11. Smyth, B. & Keane, M.T.: *Remembering to Forget: A Competence Preserving Case Deletion Policy for CBR Systems*, Proceedings of the 14th International Joint Conference on Artificial Intelligence (Mellish, C. ed.), Morgan Kaufmann, 1995, pp. 377–382. 1
12. Smyth, B. & Keane, M.T.: *Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning*, Artificial Intelligence **102** (1998), 249–293. 1
13. Smyth, B. & McKenna, E.: *Modelling the Competence of Case-Bases*, Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence (B. Smyth and P. Cunningham, eds.), Springer Verlag, 1998, pp. 208–220. 2
14. Smyth, B. & McKenna, E.: *Building Compact Competent Case-Bases*, Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence (Althoff, K-D, Bergmann, R. and Branting, L.K. eds.), Springer Verlag, 1999, pp. 329–342. 1
15. Smyth, B. & McKenna, E.: *Footprint-Based Retrieval*, Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence (Althoff, K-D, Bergmann, R. and Branting, L.K. eds.), Springer Verlag, 1999, pp. 343–357. 2
16. Zhu, J. & Yang, Q.: *Remembering to Add: Competence Preserving Case-Addition Policies for Case-Base Maintenance*, Proceedings of the 16th International Joint Conference on Artificial Intelligence, 1999, pp. 234–239. 1

# Activating Case-Based Reasoning with Active Databases

Sheng Li and Qiang Yang

School of Computing Science, Simon Fraser University  
Burnaby, BC Canada, V5A 1S6

**Abstract.** Many of today's CBR systems are passive in nature: they require human users to activate them manually and to provide information about the incoming problem explicitly. In this paper, we present an integrated system that combines CBR system with an active database system. Active databases, with the support of active rules, can perform event detecting, condition monitoring, and event handling (action execution) in an automatic manner. The combined **ActiveCBR** system consists of two layers. In the lower layer, the active database is rule-driven; in the higher layer, the result of action execution of active rules is transformed into feature-value pairs required by the CBR subsystem. The layered architecture separates case-based reasoning from complicated rule-based reasoning, and improves the traditional *passive* CBR system with the *active* property. This paper shows how to construct **ActiveCBR** system and provides an analysis of the resulting system architecture.

## 1 Introduction

As an interactive process, case retrieval in most CBR systems has been based mainly on a user-interaction model. In this model, a user provides all the information necessary for the CBR system to draw a conclusion. This '*passive*' nature of interactive CBR requires the direct involvement of human users in order to provide information about the incoming problem explicitly. This *passive mode* has some obvious drawbacks. The manual operation on information collection cannot handle massive amount of user data and real time events properly. The manual operation on system activation limits the ability to perform reasoning in timely fashion, especially in some emergent applications such as forest fire protection and coastal salvage in which the group profile of events and data trigger the operation of a CBR system.

To solve these problems, we integrate an active database system with a CBR system. An active database system is a database system that monitors situations of interest, and when they occur, triggers an appropriate response in a timely manner. An active rule in active databases extends the expert system rules with the ability of autonomously responding to external events, e.g., the modifications of data table such as *INSERT*, *DELETE*, and *UPDATE*. It generally follows the *Event-Condition-Action* paradigm.

|                            |                        |
|----------------------------|------------------------|
| <b>on</b> <i>event</i>     | # event detecting      |
| <b>if</b> <i>condition</i> | # condition monitoring |
| <b>then</b> <i>action</i>  | # action executing     |

An active database system can complement passive CBR systems in the following aspects:

- The underlying database system is capable of handling massive amount of user data, so after integrating with an active database system, CBR systems need not handle raw data directly.
- The rule mechanism is capable of detecting external events, and transferring the responses to CBR systems.
- Active database systems support complicated database query, so the statistical result of the collection of user data can be obtained.

To combine the two technologies, we propose an **ActiveCBR** architecture that builds a case-based reasoning subsystem on top of an active database, and realizes problem solving based on the data and events in a relational database. The **ActiveCBR** system developed under this architecture consists of two layers. In the lower layer, the ADB subsystem is rule-driven; in the higher-layer CB subsystem, the result of action execution of active rules is transformed into feature-value pairs required for the reasoning procedure. Connected by the rule engine in an active database, we can perform problem solving on large data sources – relational databases. Furthermore, the reasoning procedure is performed reactively in a real-time manner by responding to external events.

We can find many practical applications where the **ActiveCBR** system will provide real-time support for many industrial applications. This is because in many applications, the input data tend to repeat similar patterns from time to time. For instance, in an active database system that monitors travel information, when the season, economy and social factors recur, the interest in travel plan is likely to display regularly repetitive patterns. This observation ensures that it is reasonable to apply case-based reasoning in the problem solving in the real-time active database environment. In this paper, we motivate our research using a realistic Cable-TV diagnosis problem which we have applied our system to. In this application, the Cable-TV symptoms tend to recur with repetitions in season, user group and equipment. We show how to construct an **ActiveCBR** system using an example.

## 2 ActiveCBR: Representation and Algorithms

### 2.1 Overview of Active Databases

Active database, as a database system with reactive behavior, has been the subject of extensive research recently. The knowledge model of an active database system determines *what* can be said about active rules in the system. In contrast, the execution model indicates *how* a set of rules behaves at runtime. [5] discuss different *sources* of event that describe the happening to be monitored.

The execution of the rules depends on the event-condition and condition-action *coupling modes*, that can be *immediate*, *deferred* or *detached* [4]. The *transition granularity* describes the relationship between events and the rule execution. It can be *tuple-oriented* or *set-oriented*.

*Termination* is the key design principle for active rules. Due to the unexpected interactions between rules, termination is difficult to ensure even after a careful design. Triggering graphs are used for reasoning rules about termination. A rule set is *confluence* when any triggering of rules produce a unique final database state independent to the order of execution of the rules. A rule set guarantees *observable determinism* when all visible actions performed by rules are the same for any order of execution of the rules.

*Chimera* system [3] integrates an object-oriented data model, a declarative query language, and an active rule language. It supports *display*, *generalize*, and *specialize* events in addition to traditional *create*, *delete*, and *modify* primitives to reflect object manipulation.

Generally, the rule execution in active database systems tends to be more difficult to understand and maintain when supporting more facilities. Even in a conservative-designed rule system like Starburst [2], the semantics of rule execution are still quite complex. Rule termination and rule confluence are difficult to realize in a practical design of a large rule set. How to simplify the design and analysis of active rule sets is an important topic in active database research. With the **ActiveCBR** system, much of the user-level semantics are elevated to the CBR level, whereas the efficient rule-triggering mechanism is left to the database level. We will discuss this in detail in the next section.

## 2.2 Knowledge Representation in the ActiveCBR System

The representation of a case has various forms depending on different applications. In the **ActiveCBR** system, we define the case base as:

**Definition 1.** A case base in **ActiveCBR** system is combination of  $\{\mathcal{C}, \mathcal{F}, \mathcal{I}\}$ , where  $\mathcal{C}$ ,  $\mathcal{F}$ , and  $\mathcal{I}$  are case space, feature space, and index space, respectively.

We describe the spaces  $\{\mathcal{C}, \mathcal{F}, \mathcal{I}\}$  in above definition as follow:

- The case space  $\mathcal{C} = \{c^m \mid m = 1, \dots, M\}$  is the set of  $M$  case specifications. A specification of a case consists of *Name*, *Description*, *Threshold*, and *Solution*.
- The feature space  $\mathcal{F} = \{(f_n, v_{n,k}) \mid n = 1, \dots, N; k = 1, \dots, K_n\}$ , is the set of feature-value pairs, where  $N$  is the total number of features, and  $K_n$  is the number of possible values of feature  $f_n$ . In the case representation of the **ActiveCBR** system, all the feature values are symbolic. For the features with original numeric context, we transform them into discrete symbolic values.

- The index space  $\mathcal{I} = \{\omega(m, n, k) \mid m = 1, 2, \dots, M; n = 1, \dots, N; k = 1, \dots, K_n\}$  is the set of feature-value weights. A weight is a real number between 0 and 1<sup>1</sup>. Therefore, we can consider  $\mathcal{I}$  as such a relation:

$$\mathbf{R} : \mathcal{C} \times \mathcal{F} \rightarrow [0, 1]$$

Having the definition of the case base, we can represent a case by two parts: the specification part from an element in  $\mathcal{C}$  that describes the name, description, threshold, and solution of the case; and the index part from all the elements in  $\mathcal{I}$  that related to this case, which describe the similarity property of the case.

*Threshold* is introduced into the **ActiveCBR** system as a new field in case specification. It represents the minimum score to which the case is detected at runtime and should be fired accordingly. We will discuss the similarity and score computation in the next subsection.

*Examples* An example case in AI-CBR’s travel agents domain<sup>2</sup> is shown in Table 1.

**Table 1.** A sample case of the travel agents domain

| <b>Name</b>        |               | TravelCase31               |
|--------------------|---------------|----------------------------|
| <b>Description</b> |               | #245                       |
| <b>Threshold</b>   |               | 85                         |
| <b>Solution</b>    |               | Hotel Golden Coast, Attica |
| <b>Feature</b>     | <b>Value</b>  | <b>Weight</b>              |
| JourneyCode        | 649           | 0                          |
| Price              | \$1,000-2,499 | 80                         |
| HolidayType        | Recreation    | 35                         |
| NumOfPerson        | 1-2           | 70                         |
| Region             | Germany       | 75                         |
| Transportation     | By plane      | 45                         |
| Duration           | 5-7 days      | 85                         |
| Season             | Summer        | 65                         |
| Accommodation      | Luxury        | 70                         |

The travel-agents’ case base is used to help travel agents to recommend hotel destination for customers based on their individual interest and requirement. The solution of each case is a hotel destination. The similarity property of the example case is described by the nine feature-value-weight triples. Note that all the feature values are symbolic. Some features, such as *Price*, *Duration*, and *NumOfPerson* have original numeric values, but they have been transformed to symbolic representation. In this case base, each feature has only one value with

<sup>1</sup> In the internal representation of the **ActiveCBR** system, the weight are converted to integers between 0 and 100, and the threshold in case specifications is defined as an integer between 0 and 100 as well.

<sup>2</sup> The travel agents case base is downloaded from AI-CBR’s case-base archive. (<http://www.ai-cbr.org/cases.html>)

a positive weight. Feature *JourneyCode* is used for indexing purpose only and no positive weight is assigned. (We have omitted other feature-value pairs with zero weight in the table.)

Another example case is from a cable TV domain used by a cable TV company (shown in Table 2). In this domain, a case can have multiple positive weights for different values on a particular feature. For instance, both values ‘no picture’ and ‘reception’ of the feature ‘*ProblemType*’ are related to the case ‘*Regional switch (LB) problem*’, but the former has a higher possibility, so it is assigned with a higher weight. We will further discuss the meaning and usage of the feature weights in Section 3.

**Table 2.** A sample case of the cable TV domain

| <b>Name:</b> Regional switch (LB) problem                 |               |        |
|-----------------------------------------------------------|---------------|--------|
| <b>Description:</b> Low band regional switch is breakdown |               |        |
| <b>Threshold:</b> 78                                      |               |        |
| <b>Solution:</b> Generate ticket for technician           |               |        |
| Feature                                                   | Value         | Weight |
| ProblemType                                               | no picture    | 75     |
|                                                           | reception     | 65     |
|                                                           | VCR problem   | 0      |
| Channels                                                  | lower band    | 80     |
|                                                           | upper band    | 0      |
|                                                           | US channel    | 0      |
| Duration                                                  | recent 24 hrs | 70     |
|                                                           | recent 1 week | 45     |
|                                                           | not specified | 0      |
| Location                                                  | particular    | 85     |
|                                                           | not specified | 0      |

*Rule Representation* The representation of an active rule in the ADB subsystem depends on its underlying RDBMS. Both Oracle and SQL Server use triggers to perform the rule mechanism. A trigger in ADB subsystem is a special kind of stored procedure that is executed automatically when the specified data-modification occurs on the specific table. One trigger can contain one rule or several rules raised by the same event. A rule can be an ECA rule with complete event-condition-action semantics, or an E-A rule, in which the condition is implicitly specified by the database query language in a trigger.

The creation of a trigger is shown in Table 3. In both contexts, the *trigger-event* could be one of the three data manipulation operations, *INSERT*, *UPDATE* and *DELETE*.

*An Example* Table *ACBR\_TRAVEL\_DATA* stores the user data of the travel agent domain. The attributes of *ACBR\_TRAVEL\_DATA* are one-to-one mapping

**Table 3.** Trigger representation in SQL Server and Oracle

| SQL Server                                                                                                                                             | Oracle                                                                                                                                                                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CREATE TRIGGER</b> <i>trigger-name</i><br><b>ON</b> <i>table-name</i><br><b>FOR</b> <i>trigger-event</i><br><br><b>AS</b> <i>Transact-SQL block</i> | <b>CREATE TRIGGER</b> <i>trigger-name</i><br><b>BEFORE</b>   <b>AFTER</b> <i>trigger-event</i><br><b>ON</b> <i>table-name</i><br><b>[FOR EACH ROW</b><br><b>[WHEN (condition)]]</b><br><i>PL/SQL block</i> |

to the features of the travel agent case base. The distinction between the two is the raw data in user table can be either symbolic or numeric, e.g., attribute *price* can be any positive real number, while the value of feature *Price* is generalized and converted to symbolic value like *high* or *over 8,000*.

Consider an example rule: *If the new record has price over USD8,000, set the Price of FEATURE table to 'high'*. It can be represented as an SQL Server trigger:

```

CREATE TRIGGER INSERT_TGGR
ON ACBR_TRAVEL_DATA FOR INSERT
AS
BEGIN
    /* other rules */
    :
    IF (new.price > 8000)
        UPDATE ACBR_TRAVEL_FEATURE SET Price = 'high'
    :
    /* other rules */
END

```

Note that multiple rules are generally stored in one trigger for the INSERT event. In Oracle, it is possible to map one rule to one trigger:

```

CREATE TRIGGER PRICE_HIGH_TGGR
AFTER INSERT ON ACBR_TRAVEL_DATA
FOR EACH ROW WHEN (new.price > 8000)
BEGIN
    UPDATE ACBR_TRAVEL_FEATURE SET Price = 'high'
END

```

### 3 Algorithms

Figure 1 depicts a high-level view of the two-layer ActiveCBR architecture. Briefly, the procedures of case-based reasoning, such as case retrieval, case adap-

tation, and case maintenance, are performed in the higher *CBR Layer*; while the lower *Active Layer* encapsulates the reactive functionality to monitor the alterations of external data sources. The interaction between the two layers is carried through the feature-value pairs that are accessible to both layers. In the higher layer, feature-value pairs are used to describe the similarity between the new problem and the retained cases; while in the lower layer, they reflect the result of data alterations.

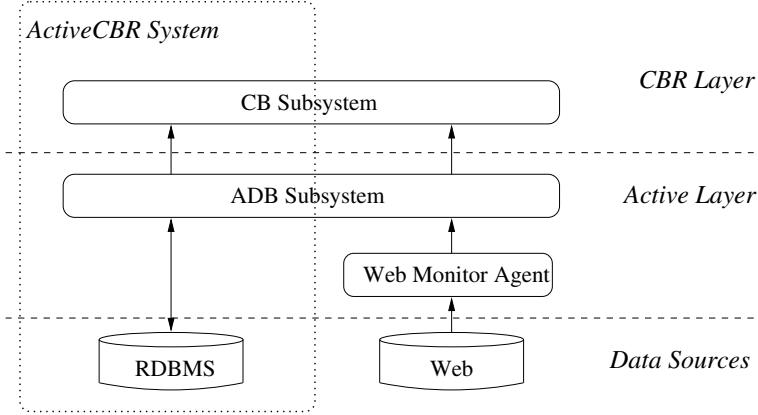


**Fig. 1.** The ActiveCBR architecture

The algorithms in the two subsystems of the **ActiveCBR** systems are independent. The *Case Authoring* module in the CB subsystem and the *Rule Definition* module in the ADB subsystem primarily work on system reconfiguration for system flexibility. We will mainly, in this section, discuss the algorithms in *Case Firing* module and *Rule Execution* module.

### 3.1 Algorithm in the CB Subsystem

Suppose we have  $M$  cases in the case base  $\mathcal{C} = \{c^1, \dots, c^M\}$ . For the  $N$  features  $f_1, \dots, f_N$ , let  $f_n^m$  denote the value of the  $n^{th}$  feature of the  $m^{th}$  case, and  $f_n^I$  denote the current input value of the  $n^{th}$  feature. Now we have:

#### Case Firing Algorithm:

1. For each new case added at runtime, mark it as enabled;
2. Retrieve current feature values  $f_n^I$ ;
3. For each case  $c^m$  in  $\mathcal{C}$  that is marked enabled:
  - (a) For each feature  $f_n$ :  
Calculate the similarity  $\text{sim}(f_n^m, f_n^I)$ ;
  - (b) Calculate the score of case  $c^m$ ;



- (c) Mark  $c_m$  as fired, if the score of  $c^m$  is greater than its threshold;
- (d) Update the firing history log, if necessary;
- 4. Visualize case firing monitor.

CBR subsystem provides a user interface to add a new case and change the enabled / disabled status of an existing case at runtime. Before the case retrieval iteration, Step 1 of case firing algorithm examines whether a new case has been added into case base, and if so, marks it as enabled. This operation maximizes the system flexibility to perform real-time knowledge management.

Let  $\kappa$  be the average number of possible values of each feature. The total number of feature-value weights is:

$$|\mathcal{I}| = M * N * \kappa \quad (1)$$

Therefore, the complexity of above case-firing algorithm is  $O(MN\kappa)$ , i.e., the algorithm is linear in terms of number of cases, number of features, and average number of values for each feature.

The linear algorithm can be improved if we have an appropriate approach to cluster the cases [10]. If the clustering is effective, we can guarantee that no case from different clusters can be fired simultaneously. Hence, we need not traverse the whole case base in the case firing algorithm; alternatively, we need just traverse along the clustering path until entering a cluster, and check the firing condition with the cases in this cluster only. On the best case, we can suppose the clustering is even and the search tree is balanced. If, on average, each cluster contains  $\lambda$  cases, we can improve the computation to:

$$O(\lambda * \log(\frac{M}{\lambda}) * N * \kappa) \quad (2)$$

### 3.2 Algorithm in the ADB Subsystem

The algorithm in the ADB subsystem depends on the trigger property of the underlying relational database. We implement the **ActiveCBR** system based on two RDBMS: SQL Server and Oracle.

The rules in Oracle support two distinct granularities, *row-level* and *statement-level*, corresponding to the instance-oriented semantics and the set-oriented semantics, respectively. They also can be executed either *before* or *after* the triggering operation. Thus, there are four possible combinations by combining the two granularities and the two evaluation times, i.e., *before* and *after* [9]. Besides the rules, database built-in integrity checking is also executed when a database manipulation occurs.

The rule processing algorithm in Oracle is:

#### Oracle Rule Processing Algorithm:

1. Execute the statement-level before-rules;
2. For each row in the target table:
  - (a) Execute the row-level before-rules;

- (b) Perform the modification of the row and row-level referential integrity and assertion checking;
- (c) Execute the row-level after-rules;
- 3. Perform the statement-level referential integrity and assertion checking;
- 4. Execute the statement-level after-rules.

## 4 Example of ActiveCBR System in Operation

In this section, we give a comprehensive example in the cable TV domain to demonstrate how the **ActiveCBR** system works.

The user data is stored in data tables such as *ACBR\_CABLE\_DATA*. The content of *ACBR\_CABLE\_DATA* is modified at runtime, e.g., insertion when new problem reported, updating when the problem solved later, and deletion when record out-of-date. Consider that two new tuples with problem id (pid) 6742 and 6743 are inserted into *ACBR\_CABLE\_DATA* (marked with \* in Table 4). An *INSERT* event occurs accordingly.

**Table 4.** User data of cable TV domain

| pid   | uid | active | type       | channel | time        | location | solved |
|-------|-----|--------|------------|---------|-------------|----------|--------|
| 6732  | 263 | Y      | No picture | 6       | 10/05 19:06 | Burnaby  | N      |
| 6733  | 546 | Y      | Reception  | all     | 10/13 12:51 | N.Van    | Y      |
| 6734  | 649 | Y      | VCR        | 15      | 10/17 21:34 | N.Van    | N      |
| 6735  | 032 | Y      | VCR        | n/a     | 10/19 02:25 | Burnaby  | N      |
| 6736  | 382 | N      | Reception  | 50      | 10/19 21:45 | Burnaby  | N      |
| 6737  | 234 | Y      | Reception  | all     | 10/20 16:23 | W.Van    | N      |
| 6738  | 271 | Y      | Reception  | 9       | 10/20 20:42 | Burnaby  | N      |
| 6739  | 031 | Y      | No picture | 13      | 10/20 22:19 | Burnaby  | N      |
| 6740  | 740 | Y      | VCR        | 11      | 10/20 22:43 | UBC      | N      |
| 6741  | 638 | Y      | VCR        | 28      | 10/20 23:19 | SFU      | N      |
| 6742* | 957 | Y      | Reception  | 3       | 10/20 23:32 | Burnaby  | N      |
| 6743* | 271 | Y      | Reception  | 6       | 10/20 23:57 | Burnaby  | N      |

The preprocessing performs slicing and dicing operations on user data to reduce the size of query table. For instance, in slicing operation, the tuples with 'N' value of *active* attribute are excluded from further query; in dicing operation, all the attributes that are not related to rule conditions such as *pid* and *solved* are removed as well. A temporary table #USERDATA is created in the preprocessing, and the number of tuples in #USERDATA is counted into variable @num.

```

SELECT type, channel, time, location
INTO #USERDATA /* user data after preprocessing */
FROM ACBR_CABLE_DATA
WHERE active = 1
SELECT @num = COUNT(*) /* number of tuples */
FROM #USERDATA

```

Next, rule conditions are evaluated upon the *#USERDATA* table. In the *ActiveCBR* system, the values of features of the higher level case base are updated dynamically by the rule action from the lower level active database. For each feature in the feature space, there are several active rules to be evaluated, but only one of the conditions could be true, then the feature is set to a corresponding value.

Consider the case in Table 2. For feature *Duration*, its value can be ‘*recent 12 hrs*’, ‘*recent 3 days*’, and ‘*not specified*’. The rule used to update feature *Duration* to ‘*recent 12 hrs*’ can be described as:

```

on   INSERT
if   At least 1/3 of total tuples and at least 5 tuples
       are reported within the last 12 hours.
then Update feature Duration with value ‘recent 12 hrs’.

```

The rule can be represented as a block of SQL statements in the *INSERT* event trigger:

```

:
SELECT @npart = COUNT(*)
FROM #USERDATA
WHERE DATEDIFF(hour, time, GETDATE()) < 12
IF ((@npart * 3 > @num) AND (@npart > 5))
BEGIN
    UPDATE ACBR_CABLE_FEATURE
    SET value = ‘recent 12 hrs’
    WHERE feature = ‘Duration’
END
:

```

After all rules are executed, we have the runtime feature values as shown in row *Current Value* in Table 5.

The result of similarity computation is listed in the last row in Table 5. Accordingly, we can compute the score of this case. In the above example, the result score is 96. Since it is greater than the threshold 78, the case *Regional switch (LB) problem* is marked as ‘*fired*’.

**Table 5.** Runtime feature values and similarity computation of case *Regional switch (LB) problem*

|                                               | ProblemType | Channels   | Duration      | Location   |
|-----------------------------------------------|-------------|------------|---------------|------------|
| Current Value<br>$f_n^I$                      | reception   | lower band | recent 12 hrs | particular |
| Importance Weight<br>$w_n^m$                  | 75          | 80         | 70            | 85         |
| Feature Value Weight<br>$\omega(n, m, f_n^I)$ | 65          | 80         | 70            | 85         |
| similarity<br>$sim(f_n^m, f_n^I)$             | 0.87        | 1.00       | 1.00          | 1.00       |

The system will fire and visualize every case whose score exceeds its threshold at runtime.

## 5 Conclusions and Future Work

This work is originally based on the need to improve traditional ‘passive’ case-based reasoning system to have active property. The ‘active’ property has two related meanings: first, the system should be capable of responding to an external event. The response should be made in real-time fashion within a limit of time.

The contributions of this work are two-fold. The combined knowledge representation in different level improves system performance by reducing the size of rule set and the confliction between the rules by factoring out the two sets of knowledge bases. Also, the combined system allows case-based reasoning to apply to real-time database environment.

A prevailing trend of distributed computing demands a distributed architecture for our **ActiveCBR** applications. Several models are under consideration, such as using one CB subsystem via multiple ADB subsystems or multiple CB subsystems being centralized with a predominant CB system. To complete distributed tasks effectively, the **ActiveCBR** system should be enhanced to have capacity of integrating multiple-source information, performing inter-subsystem communication, and avoiding possible information bottleneck.

## Acknowledgment

We thank NSERC and IRIS for their support for this research.

## References

1. A. Aamodt and E. Plaza. Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1993.

2. A. Biliris. The performance of three database storage structures for managing large objects. In *ACM SIGMOD Conference on the Management of Data*, San Diego, CA, 1992. 5
3. S. Ceri, P. Fraternali, S. Paraboschi, and L. Branca. Active rule management in Chimera. In *Active Database Systems - Triggers and Rules For Advanced Database Processing*, pages 151–76. Morgan Kaufman, 1996. 5
4. O. Diaz and A. Jaime. EXACT: an extensible approach to active object-oriented databases. *VLDB Journal*, 6(4):282–295, 1997. 5
5. O. Diaz, A. Jaime, N. W. Paton, and G. Qaimari. Supporting dynamic displays using active rules. *ACM SIGMOD Record*, 23(1):21–26, 1994. 4
6. D. Gentner. Structure mapping: a theoretical framework for analogy. *Cognitive Science*, 7:155–70, 1983.
7. J. L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publisers, Inc., 1993.
8. D. B. Leake, A. Kinley, and D. Wilson. Learning to improve case adaptation by introspective reasoning and CBR. In *Proceedings of the First International Conference on Case-Based Reasoning*. Springer-Verlag, 1995.
9. K. Owens and S. Adams. Oracle 7 triggers: Mutating tables? *Database Programming and Design*, 7(10):31–49, 1994. 10
10. J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 10
11. Y. Shoham. An overview of agent-oriented programming. In J. M. Bradshaw, editor, *Software Agents*, pages 271–90. AAAI Press, 1997.
12. I. Watson and F. Marir. Case-based reasoning: A review. *The Knowledge Engineering Review*, 9(4):355–81, 1994.

# Case-Based Reasoning with Confidence

William Cheetham (cheetham@crd.ge.com)  
phone: (518) 387-5222 fax: (518) 387-6845,  
GE Research and Development Center,  
1 Research Circle, Niskayuna, NY 12309

**Abstract.** A case-based reasoning system can produce both a solution and an estimate of the confidence in that solution. The confidence value can be used to determine whether the solution does or does not have the needed accuracy. A statistical method can be used to compute a confidence value from information generated during the case-based reasoning process. This confidence value allows users to know when the results of the system should and should not be used.

## 1 Introduction

Can you always believe the results of a case-based reasoning (CBR) system? Imagine you are a doctor making a life-or-death diagnosis of a person with possible heart failure based on the output of the CASEY [10] CBR system or an executive planning a corporate merger or acquisition using the MARS [2] CBR system. In either of these cases, and many others, you would have a need for understanding when the CBR system does and does not produce an accurate solution. A single CBR system could have many applicable cases for one problem and few for another problem. This paper describes a methodology that allows a CBR system to provide both a solution and its confidence in that solution, as in Figure 1.

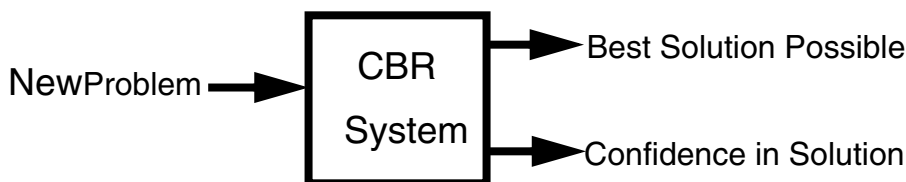


**Figure 1: CBR with Confidence**

This paper will provide background on the need for confidence in a CBR system. Then, a method of determining confidence will be presented. Finally, results for two test domains will show the usefulness of the confidence value.

## 2 Background

Current CBR systems do not provide an explicit measure of the confidence in their solution. Three ways that current systems give some impression of the confidence in the solution are

- displaying the retrieved cases
- displaying the similarity scores for the retrieved cases, and
- simulating the solution

Displaying the retrieved cases to a user allows the user to make their own judgement on the confidence that should be placed on the solution. If the retrieved cases seem to support the solution then the solution can be taken with more confidence than if the retrieved cases do not seem to support it. This method has some serious drawbacks. No CBR system provides a procedure for determining confidence from a set of cases. This means that different users can have different methods for determining confidence and that there is no procedure that can be automated. Furthermore, it is not always true that the user will be an expert in the area of the problem that is solved by the CBR system. These drawbacks make displaying the retrieved cases an imprecise method for determining confidence.

Displaying the similarity scores that were calculated during retrieval is another method of providing a confidence. The idea is that the higher the similarity score is, the higher the confidence should be. However, the similarity measure simply rates the similarity of the surface features of the case and the subject. A similarity in surface features does not guarantee that the case will be useful with regard to the current subject. Up until now, similarity scores have only been used to rank cases within a single retrieval. A retrieved case with a higher similarity score is presumed to be better than a case with a lower similarity score. This says nothing about the quality of a particular similarity score. For example, it is unknown if a similarity score of 0.6 (on a scale from 0 to 1) should be taken as having confidence in a solution or not. More meaning needs to be placed in a similarity score before such a determination can be made.

A final method for generating a confidence is simulating a solution. Simulations have been done in two ways; using a software simulation to test the solution (as was done in the recipe generating system CHEF [7]), or actually testing the solution in the real world, possibly on a smaller scale. However, the software simulations are often as error prone as creating the solution, and testing the solution in the real world has many practical drawbacks in that it can not be automated, does not always scale up, and can be expensive.

### 3 Method for Determining Confidence

The goal of determining confidence is to produce a formula that takes information about the execution of a CBR system on a specific subject and calculates the confidence that the solution generated for that subject is correct. We have implemented and experimented with multiple methods for determining confidence in a CBR system. These methods combine and extend ideas from areas including statistics [5], cluster analysis [6], fuzzy logic [8, 11], and CBR similarity metrics [1, 9]. One technique will be described in this paper.

The correctness of a CBR systems depend on many properties of the CBR system. The confidence measures use quantitative values for information that is created during the CBR process such as

- similarity scores of the cases retrieved,
- the typicality of the subject with respect to the case-base,
- the deviation in solutions suggested by the retrieved cases,
- percentage of cases retrieved that suggest a specific solution,

- span of all solutions suggested, ...

The confidence methods turn these “raw” sources of data into a specific confidence scale. Different information is important for different CBR systems. The method for determining the confidence in a given CBR system can be performed for each type of information that is applicable to the specific CBR process. Then these individual confidence measures can be combined to produce a single value which is the confidence.

The first thing that needs to be done in creating confidence measures is to determine a confidence scale (number of confidence intervals and allowable error for each interval). Setting the confidence scale specifies the performance needed from the CBR system. Each interval corresponds to an amount of error that should be expected in the solution. The error in a solution should be defined in a way that a set of training cases can have the error determined for each case. This error can be determined by whatever method is most convenient for that application. A few examples are:

- if the solution was correct or not (0 or 1)
- the percentage error in the calculated vs. actual solutions [0, 1]
- an expert can rate the solution to each example [0, 1]

We created intervals that have names that correspond to fuzzy linguistic values. An example of a confidence scale with three confidence intervals is given in Table 1

**Table 1: Confidence Scale**

| Fuzzy Linguistic Term | Confidence Interval | Average Error      |
|-----------------------|---------------------|--------------------|
| good                  | 0.75 - 1            | less than 5%       |
| questionable          | 0.5 - 0.75          | between 5% and 10% |
| poor                  | 0 - 0.5             | greater than 10%   |

After the confidence scale has been set, an analysis of a training set is used to determine error and confidence typical for specific values of the information that is generated during the CBR process. In the remainder of this section, the similarity scores generated is used as an example of this information. The following process was also informative in suggesting ways to improve or add meaning to the similarity calculation.

In order to create a confidence measure based on similarity, the similarity score and error in the solution should be recorded for each training subject. A large enough training set must be used so that the statistical confidence interval can be determined for the data. A piecewise linear function for determining typical error based on similarity will first be created. Then the a method for converting similarity to confidence will be determined.

In order to construct the linear or piecewise linear regression that fits the data, we used the package S-PLUS [13] to first fit a nonlinear regression to the data. The endpoints of the piecewise linear section were obtained from finding the endpoints of the straight portions of this nonlinear regression. After the piecewise linear regression is constructed, we determine the  $(1 - \alpha) * 100\%$  confidence interval for the data.  $\alpha$  is the percentage that is outside the confidence interval. Figure 2 shows piecewise linear set



of line segments between the upper and lower confidence limits of the confidence inter-

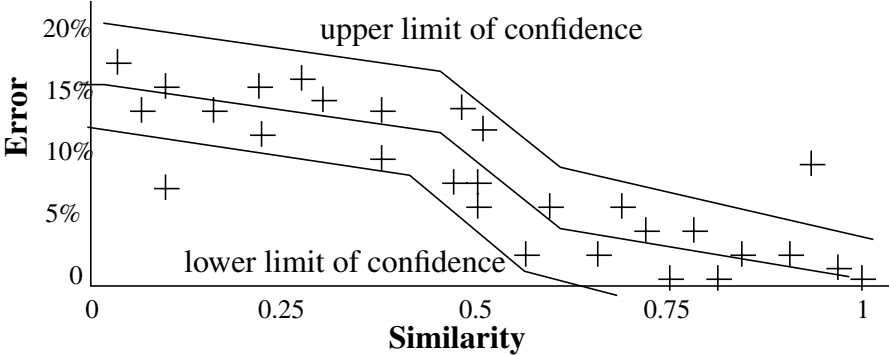


Figure 2: Confidence Interval

val. The confidence interval will let us know what range of error is likely for each similarity score. The statistical formula that can be used to generate them is given in chapter 7 of [14].

From the definition of the confidence intervals and relationship between similarity and error we can construct a mapping from similarity to confidence. The lowest similarity score that satisfies the error conditions for a confidence interval with  $(1-\alpha) \cdot 100\%$  confidence is used as the lower threshold for that interval. These thresholds form a piecewise linear function to calculate confidence from similarity. For the example of Figure 2, where good confidence means error less than 5%, the lowest similarity that has error 5% is 0.85. The lowest similarity with error 10% is 0.6. The highest similarity with error above 10% is 0.4. Then the points [0.85, 0.75] (low similarity for good, low confidence for good), [0.6, 0.5] (low similarity for questionable, low confidence for questionable) [0.4, 0] (high similarity for poor, low confidence for poor) can be used to construct the transformation from similarity to confidence. A graph of these points, called a fuzzy preference function, is given in Figure 3.

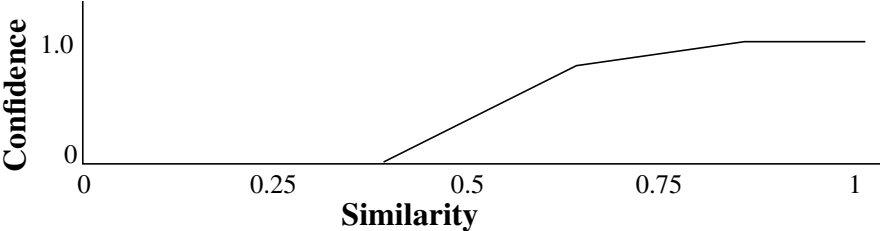


Figure 3: Individual Confidence Measure for Similarity

A 95% confidence threshold is usually preferred by statisticians, but lower confidence thresholds could be created. Multiple thresholds can be compared and the one that allows the best classification of high confidence solutions having low error should be selected. A higher confidence threshold will classify fewer solutions incorrectly, but a lower threshold will allow more solutions to be classified with high confidence. This is a trade-off where the best solution depends on the characteristics of the domain.

Multiple individual confidence measures can be combined into a single confidence measure because they all use the same confidence scale. Two ways to combine the individual confidence measures are using weighted aggregation or by taking the minimum of their values.

## 4 Test Results

The methodology described above has been applied to two CBR domains, property valuation (determining a dollar value for residential real estate properties) [3] and plastics color matching (finding a set of colorants that match a color standard under all lighting conditions). Both of these domains have traditionally been done in a non-automated way that is similar to the method of CBR. Property valuation has been done by finding similar properties that had been sold recently and using the value of these properties (which acted as cases) to determine a value for the property. Color matching was done by having a filing cabinet filled with examples of previous colors that had been matched (which acts as a case-base). When a new color was to be matched the filing cabinet would be searched for the most similar previous color sample. The pigments and loadings used for the most similar sample from the filing cabinet would be used as a starting point for the color match.

In addition to being appropriate CBR domains, these domains also show two different types of CBR. The property valuation domain takes multiple cases and combines these cases into a single solution. The color matching domain takes the single best case and adapts that case into a better solution. Together, these domains represent an appropriate and varied set of test domains. Each of the implementations of CBR using confidence will be described.

### 4.1 Confidence in Property Valuation

This section will show how the statistical measures of confidence, described in the previous section, were applied to the property valuation domain. A case-base that contains over 100,000 cases was used for the testing. Each case represented a recently sold property in the San Francisco area. A training set of 7,293 properties was used as the subjects in the training. For each of the 7,293 properties, the predicted sale price was calculated and compared with its actual sale price, using formula 1, in order to calculate

$$\text{Error} = \frac{\text{Actual\_Sale\_Price} - \text{Predicted\_Sale\_Price}}{\text{Actual\_Sale\_Price}} \quad (1)$$

the error in the estimate. Information that was gathered during the training, see Table 2, included the number of comparable properties found (Typicality), the average sim-

**Table 2: Property Valuation Training Data**

| Test # | % Error | Typicality | Similarity | Deviation |
|--------|---------|------------|------------|-----------|
| 1      | -29.8   | 3          | 0.629      | 12.020%   |
| 2      | -2.0    | 35         | 0.936      | 2.248     |
| 3      | 17.3    | 11         | 0.711      | 5.672     |
| 4      | 0.5     | 24         | 0.852      | 2.056     |
| 5      | -1.6    | 14         | 0.949      | 2.891     |

ilarity score of the top comparables (Similarity), and the average comparable deviation

from estimated value (Deviation). The the remainder of this section will show how a confidence measures was created using similarity. The same process was used to create individual confidence measures for typicality and deviation. Results on all three confidence measures will be discussed.

4.1.2 Creating a Confidence Measure for Similarity

This section describes how similarity is used as a confidence measure. The typical error for a given similarity score is calculated and a function that converts similarity to confidence is created.

The similarity and error were determined for each training subject. This data was plotted and the statistical package S-PLUS was used to create a nonlinear curve that fits the data. The fitted curve gave the most likely error for a given similarity. The Figure 4 shows the fitted curve and the 90% confidence intervals. Note that Figure 4 show 300

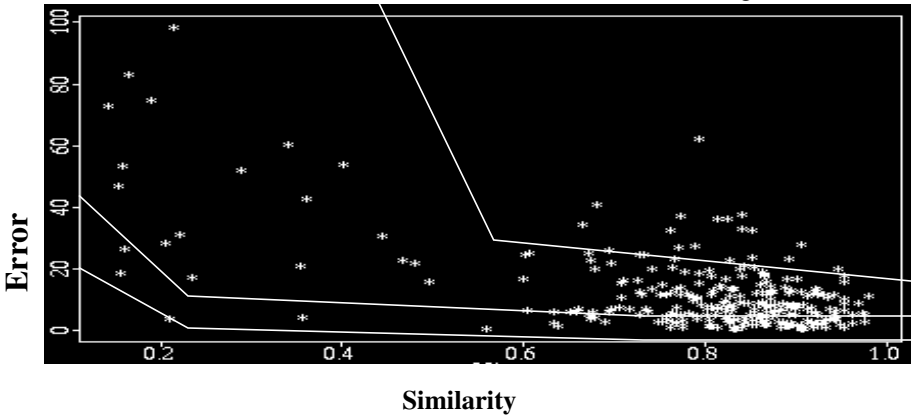


Figure 4: Similarity vs. Error

representative data points from the training set of 7,293 because the full plots had so many overlapping points that the 300 point plots were easier to read.

The information in Figure 4 was used to construct a function that transforms similarity into confidence. The criteria for good confidence was the error must be less than 10%. Subjects with similarity scores above 0.85 had error under 10% 90% of the time. Since these subjects are likely to be correct, the confidence in these subject should be very good, 1.0. The criteria for poor confidence was the error was over 20%. 90% of the

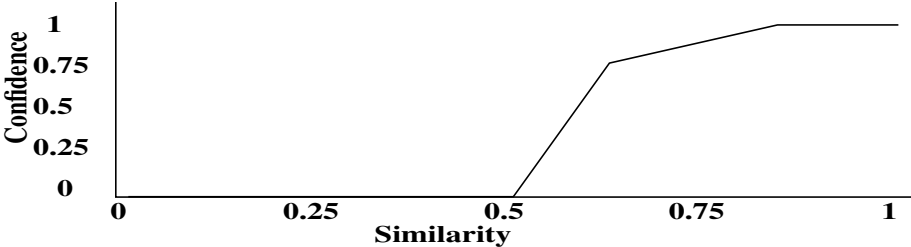


Figure 5: Individual Confidence Measure for Similarity

subjects with similarity less than 0.5 have a error that is over 20% Therefore, we con-

cluded the confidence in these should be very poor, 0. The fuzzy preference function in Figure 5 was created.

Figure 7 shows a linear fit to points that were derived in the prior analysis. This confidence is the individual confidence for similarity and is combined with other individual confidence measures to provide a complete confidence measure.

#### 4.1.1 Creating a Confidence Measure for Typicality

Judging the typicality of a subject in the property valuation domain involves finding how many other properties in the same neighborhood are similar to the subject. Another property is considered similar if its key attributes (i.e. distance, date\_of\_sale, living\_area, lot\_size, bedrooms, and bathrooms) are all within a tolerable level of difference from the subject. The tolerable level for the key attributes is that each individual attribute must have a similarity score greater than zero. The retrieval phase in our estimator first selects all comparables that have their key attributes within this tolerable difference from the subject. The number of comparables that are obtained because they are within the tolerable limit can be used to determine the confidence in the solution by the same method that was used to generate a confidence value from the similarity. Figure 6 shows this fitted curve and the 90% confidence intervals. Figure 7 shows the

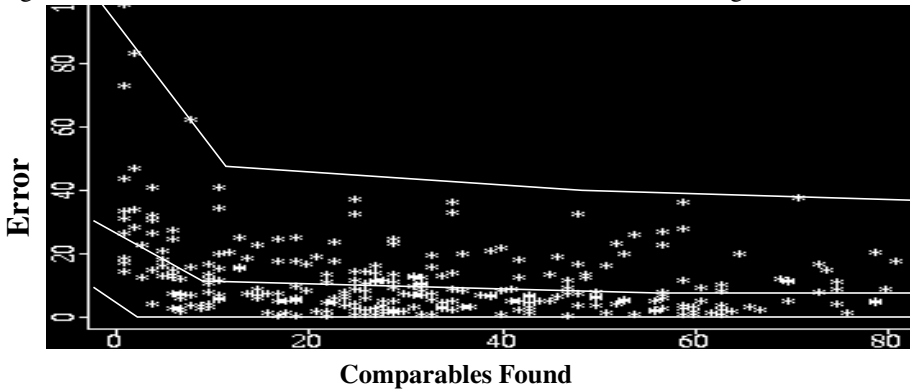


Figure 6: Comparables Found vs. Error

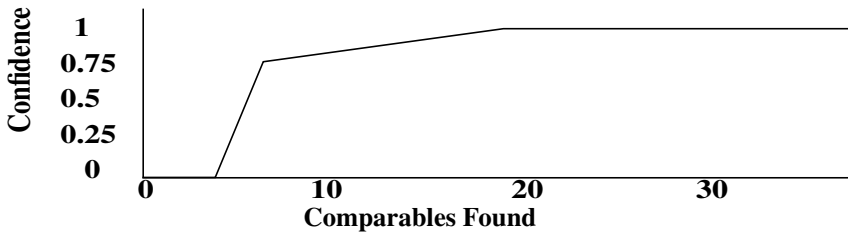


Figure 7: Individual Confidence Measure for Comparables Found

fuzzy preference function that was created from figure 6.

The typicality analysis assumes that the distribution of cases in the case-base reflects the actual occurrence of cases. In domains where only prototypical cases are retained (i.e. all duplicate cases are deleted) or abstractions are created when cases are similar (i.e. cases are combined together to form a single case), a count of the number of cases represented by the prototypical or abstract case should be maintained.

4.1.2 Creating a Confidence Measure for Deviation of Cases

We can have a greater confidence in an estimated value for a subject if the comparables are all suggesting the same estimate than if each comparable is suggesting a different value for the estimate. This individual confidence measure is based on the deviation in values estimated by the comparables. The deviation is calculated by determining the percentage difference in the estimated price of each comparable selected with the estimated value of the subject. For example, Figure 8 shows this calculation for four com-

| Comparable        | Adj. Price - Estimate | Percent deviation |
|-------------------|-----------------------|-------------------|
| comparable 1      | 110,000 - 100,000     | 10%               |
| comparable 2      | 90,000 - 100,000      | 10%               |
| comparable 3      | 105,000 - 100,000     | 5%                |
| comparable 4      | 95,000 - 100,000      | 5%                |
| Average Deviation |                       | 7.5%              |

Figure 8: Confidence Aggregation

parables, where the estimated value of the subject is \$100,000. The average percentage deviation is the value that represents the deviation.

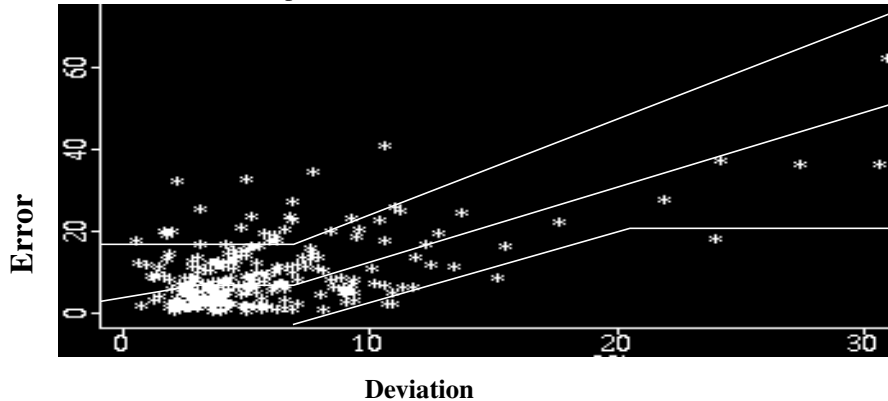


Figure 9: Deviation vs. Error

Other deviations such as sum of squared difference (standard deviation) and percentage difference in highest and lowest estimate were also tested. These other deviations did not predict the error as well as non-squared deviation because there was a significant possibility of inaccuracies in the data that was used to calculate the adjusted prices.

These other deviations would predict a large error if even one of the comparables was out of line with the others. There can be many reasons for a single comparable to be different than the others (the property needed to be sold quickly, there is value in the house that can not be represented in the attributes, etc.). A set of comparables is used to minimize the noise. The confidence measure should also minimize the misrepresentation of confidence because of imperfect data.

For each training subject the deviation and error were determined. Figure 9 shows the curve that was fitted to this data with S-PLUS and the 90% confidence intervals.

Figure 10 shows the fuzzy preference function that was created from figure 9.

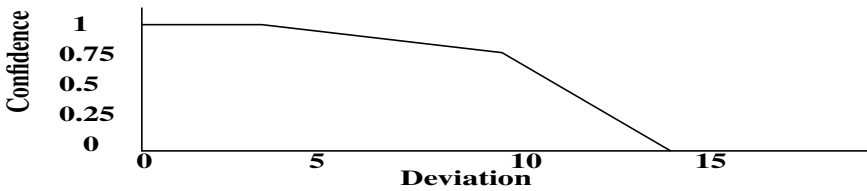


**Figure 10: Individual Confidence Measure for Deviation**

#### 4.1.3 Results

The confidence and error were calculated for each subject in the entire case-base (i.e. all 100,000). Each confidence measure was evaluated to determine its contribution to the total confidence score. The relative importance of each of the three individual confidence measures was investigated. Each individual confidence measure had statistics calculated for the percentage of subjects that the measure considers poor or questionable, and the median error of these poor or questionable subjects, see Table 3. The more

**Table 3: Comparison of Confidence Measures**

| Conf. Measure | % of subjects with low conf. | Median error of low conf. set |
|---------------|------------------------------|-------------------------------|
| Similarity    | 9.8%                         | 13.1%                         |
| Typicality    | 14.5%                        | 10.7%                         |
| Deviation     | 11.2%                        | 12.6%                         |

subjects with high error that a confidence measure can classify as having low confidence the better. So, the best confidence measure would have the highest values for both percent of subjects it can classify with low confidence and the median error of those subjects. However, the more subjects that are classified with low confidence the harder it is to find other subjects that should also be classified with low confidence.

For the entire test set, the median error was 6.4%. The confidence analysis classified 73% of the subjects as good. The good set had a median error of 4.9% and the non-

good set (i.e. poor or questionable) had a median error of 11.7%. The median error drop from 6.4% to 4.9% has greatly increased the usefulness of the CBR system.

4.2 Statistical Confidence Measures in Color Matching

The statistical confidence measures were also used for the color matching domain. 100 color matches were used as a test set for the following analysis. For this domain, confidence measures were created for similarity and typicality. The deviation among cases retrieved was not calculated because there was only one case that was retrieved. The two confidence measures were evaluated on whether they had good confidence in the test subjects that were rated good by an expert. Only the confidence measure for similarity will be described here.

The following analysis uses the similarity of the case that had the highest similarity after adaptation. The first step is to determine the relationship between the similarity score of the best case and the expert's rating. This relationship is shown in Figure 11.

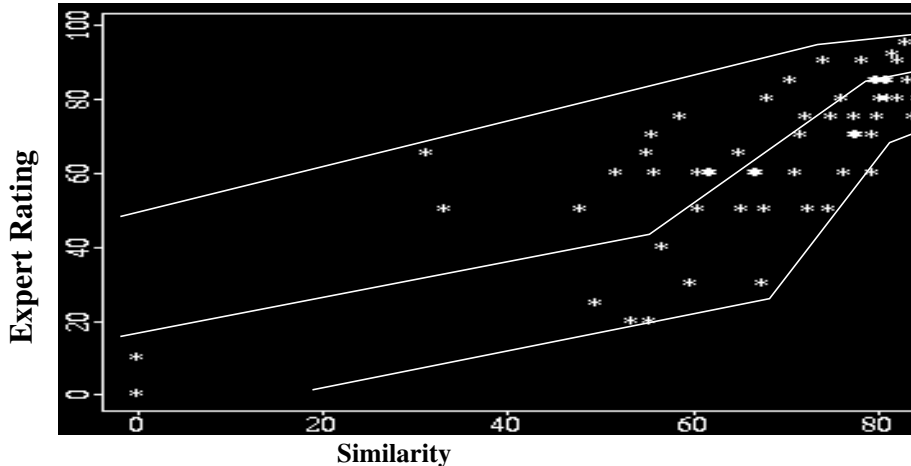


Figure 11: Similarity vs. Expert Rating

The three piecewise linear curves show the curve fitted to the data and the 95% confidence interval that we created. From this graph we constructed the confidence measure shown in Figure 12.

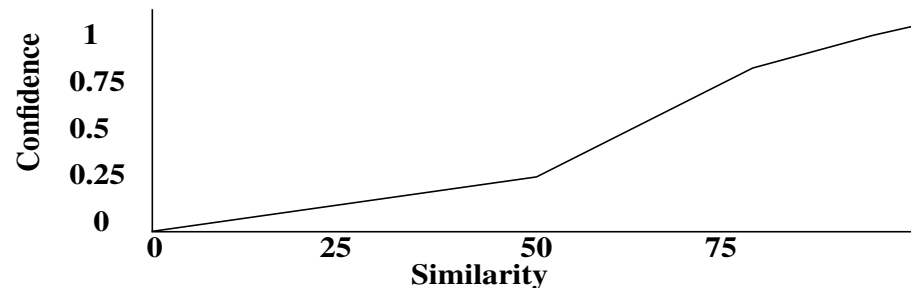


Figure 12: Individual Confidence Measure for Similarity

The individual confidence measure for similarity had good confidence in 94% of the subjects that the expert considered good. This confidence measure is now being used to determine if the color match suggested by the case-based reasoning system should be used or if a labor intensive process of color matching should be started. The expert is only needed for borderline cases.

## 5 Summary

This paper showed that the confidence measures provided a dramatic increase in the usefulness of the two systems described. Confidence measures for three types of information (similarity, typicality, and deviation) were shown to be easily created from training data. We feel that this technique can be applied to other domains with similar success.

## 6 References

- [1] Althoff, K-D., *A Review of Industrial Case-Based Reasoning Tools*, AI Intelligence, United Kingdom, 1995.
- [2] Bonissone, P., *Mars: A Mergers & Acquisitions Reasoning System*, Computer Science in Economics and Management, Kluwer Academic Publishers, Holland, 1990.
- [3] Bonissone, P., *Financial Applications of Fuzzy Case-Based Reasoning to Residential Property Valuation*, IEEE-Fuzzy 97, Barcelona, Spain, 1997.
- [4] Cheetham, W., *Case-Based Reasoning with Confidence*, Ph.D Thesis, Rensselaer Polytechnic Institute, December 1996.
- [5] Efron, B., *An Introduction to the Bootstrap*, Chapman & Hall, 1993.
- [6] Everitt, B., *Cluster Analysis*, Third Edition, John Wiley and Sons Inc., London, 1993.
- [7] Hammond, K., "Case Based Planning: Viewing Planning as a Memory Task," Academic Press, Boston, 1989.
- [8] Klir, G., *Fuzzy Sets, Uncertainty, and Information*, Prentice Hall, 1988.
- [9] Kolodner, J., *Case-Based Reasoning*. Morgan Kaufmann, 1993.
- [10] Koton, P., *Reasoning About Evidence in Causal Explanation*, In Proceedings of AAAI-88. Cambridge, MA: AAAI Press/MIT Press, 1988.
- [11] Mendel, J., *Fuzzy Logic Systems for Engineering: A Tutorial*, Proceedings of the IEEE, Vol. 83, no. 3, March 1995.
- [12] Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- [13] StatSci (A division of MathSoft), *S-PLUS Guide to Statistical and Mathematical Analysis*, Seattle, Washington, 1993.
- [14] Walpole, R., *Probability and Statistics for Engineers and Scientists*, Macmillan Publishing, New York, 1985.



# Combining Rule-Based and Case-Based Learning for Iterative Part-of-Speech Tagging

Alneu de Andrade Lopes and Alípio Jorge

LIACC - Laboratório de Inteligência Artificial e Ciências de Computadores  
Universidade do Porto - R. do Campo Alegre 823, 4150 Porto, Portugal  
alneu@ncc.up.pt  
amjorge@ncc.up.pt

**Abstract.** In this article we show how the accuracy of a rule based first order theory may be increased by combining it with a case-based approach in a classification task. Case-based learning is used when the rule language bias is exhausted. This is achieved in an iterative approach. In each iteration theories consisting of first order rules are induced and covered examples are removed. The process stops when it is no longer possible to find rules with satisfactory quality. The remaining examples are then handled as cases. The case-based approach proposed here is also, to a large extent, new. Instead of only storing the cases as provided, it has a learning phase where, for each case, it constructs and stores a set of explanations with support and confidence above given thresholds. These explanations have different levels of generality and the maximally specific one corresponds to the case itself. The same case may have different explanations representing different perspectives of the case. Therefore, to classify a new case, it looks for relevant stored explanations applicable to the new case. The different possible views of the case given by the explanations correspond to considering different sets of conditions/features to analyze the case. In other words, they lead to different ways to compute similarity between known cases/explanations and the new case to be classified (as opposed to the commonly used global metric). Experimental results have been obtained on a corpus of Portuguese texts for the task of part-of-speech tagging with significant improvement.

## 1 Introduction

Often, computational natural language processing requires that each word in a given text is correctly classified according to its role. This classification task is known as *part-of-speech tagging* and it consists in assigning to each word in a given body of text an appropriate grammatical category like noun, article, ordinal number, etc., according to the role of the word in that particular context. These categories are called

part-of-speech tags and may total a few tens, depending on the variants one considers for each particular category. The difficulty of this task lies in the fact that a given word may play different roles in different contexts. Although there is, for each word, a relatively small set of possible tags, for many words there is more than one tag. Words with a single possible tag are handled by employing a simple lookup-table (dictionary). The way to solve the ambiguity for words with more than one possible tag is by considering the context of the word and possibly employing background knowledge.

Tagging words manually is a tedious and error-prone activity, and learning approaches have been proven useful in the automation of this task. In (Jorge & Lopes 1999) an iterative approach was proposed that can learn, from scratch, a recursive first order decision list able to tag words in a text. This approach proceeds by learning a theory containing context free tagging rules in the first iteration and then by learning context dependent recursive theories in subsequent iterations until all words can be tagged. The training data employed for each iteration consists of the words that could not be tagged in previous iterations.

Despite the high predictive ability of the iterative tagging approach, it was clear that the theory produced in the last iteration was responsible for a large number of errors that would have a large impact in the overall result. These remaining cases include noise, rare exceptions or examples that cannot be expressed in the given language. Therefore, we expected that a case-based approach could cope with these residual cases better than an induced theory. However, experiments show that a traditional case-based reasoning algorithm, based on overlapping similarity measures and weighted features, does not improve the results of the rule-based approach (section 8). Besides, it is difficult to choose the appropriate weights for the features. Although typically closest neighbors are more relevant, and therefore require higher weights, it is often the case that the tag of one word is determined by other more distant neighbors. Situations like these may be common in the residual cases handled in the last iteration.

The current approach exploits the concept of case understanding to allow retrieval of good similar cases to solve it. For that, we have developed a case-based reasoning algorithm, RC2, consisting of two phases. In the learning phase, it constructs *explanations* of the cases. In the classification phase, it uses these explanations to classify new cases. Instead of using a fixed metric, each case is analyzed using an appropriate set of conditions defined by constructed explanations.

In summary, in this paper we present an iterative strategy that combines a rule based learning in the first iterations with a case-based learning in the last one. This paradigm shift during learning overcomes the limitations of the rule language and increases the predictive accuracy in the last iteration and overall. The approach proposed is new and will be described in some detail.

## 2 The Problem

A sentence is a sequence of words for which we are interested in assigning an appropriate tag for each of its constituents. The tag is an appropriate grammatical

category. Usually, from the original sequence we take a part of it (a window) with some number of elements to the left and to the right of a given position, and we try to predict the value related to the element in the central position; in general employing background knowledge too.

In the task of Part-of-speech tagging the assignment must take into account the role of the word in that particular context. The difficulty of this task lies in the fact that a given word may play different roles in different contexts.

Below we show two sequences representing two sentences.

|       |     |     |    |     |     |    |      |     |     |     |
|-------|-----|-----|----|-----|-----|----|------|-----|-----|-----|
| Words | The | car | is | red | .   | I  | like | the | car | .   |
| Tags  | art | n   | v  | adj | dot | pr | v    | art | n   | dot |

We start by representing the text or the primary sequence of words to be tagged as a set of facts.

```
{word(s1,1,'The'), word(s1,2,car), word(s1,3,is), word(s1,4,red),
word(s1,5,'.'), word(s2,1,'I'), word(s2,2,like), word(s2,3,the),
word(s2,4,car), word(s2,5,'.' ) }
```

In the above example, *s1* and *s2* are sentence labels. The second argument is the position of the word within the sentence. Punctuation marks such as "." are regarded as words. The corresponding tags are also represented as facts.

```
{tag(s1,1,art), tag(s1,2,noun), tag(s1,3,v), ... , tag(s2,5,dot)}
```

### 3 The Iterative Induction Strategy

The iterative approach to part-of-speech tagging presented in Jorge & Lopes (1999) tackles the problem of learning recursive first order clauses and is mainly based on work done in the context of inductive program synthesis (Jorge & Brazdil 1996) and (Jorge 1998).

In this approach, we start by inducing clauses that are able to determine the tag of some of the words in a given set, without any context information and with confidence above of a given threshold. These are the first clauses to be applied in the classification. They are the base clauses of the recursive definition we want to induce and are not recursive. These clauses are also used to enrich the background knowledge, thus enabling and/or facilitating the synthesis of recursive clauses in the following iterations.

Having obtained this first layer of clauses, let us call it  $T_1$ , we are able to classify (tag) some of the words in the text used for training. Using the answers given by this theory  $T_1$  we may induce some recursive context clauses thus obtaining theory  $T_2$ . By iterating the process, we obtain a sequence of theories  $T_1, T_2, \dots, T_n$ . The final theory is

$$T = T_1 \cup T_2 \cup \dots \cup T_n.$$

To induce each theory in the sequence we may apply a sort of covering strategy, by considering as training examples in iteration  $i$  only the ones that have not been covered by theories  $T_1, \dots, T_{i-1}$ . We stop iterating when all the examples have been

covered, or when we cannot find any clauses. To handle the remaining examples, we consider all clauses for selection regardless of their confidence (quality).

The construction of each theory  $T_1, T_2, \dots$  is done by a given learning algorithm. In this article the learning algorithm ALG used is the first order rule inducer CSC(RC1), for all but the last iteration. There we employ a case-based reasoning strategy.

---

**Algorithm 1:** Iterative Induction

---

**Given**

Language  $L$ , examples  $E$  and background knowledge  $BK$ ,  
 Confidence level  $C$   
 Learning algorithm  $ALG(E, BK, C)$

**Find**

A theory  $T$  in  $L$

**Algorithm:**

```

Uncovered  $\leftarrow E$ ,  $T \leftarrow \emptyset$ ,  $i \leftarrow 1$ 
Do
     $T_i \leftarrow ALG(Uncovered, BK, C)$ 
     $T \leftarrow T \cup T_i$ 
     $BK \leftarrow BK \cup T_i$ 
     $Uncovered \leftarrow Uncovered \setminus covered\_examples(T_i)$ 
     $i \leftarrow i + 1$ 
Until  $covered\_examples(T_i) = \emptyset$  or  $Uncovered = \emptyset$ 
 $T \leftarrow T \cup ALG(Uncovered, BK, 0)$ 
  
```

---

**Example:** Assume that the background knowledge includes the definition of the predicate `word/3` (describing the text) and `window/9` defined as

```

window(P, L1, L2, L3, L4, R1, R2, R3, R4)  $\leftarrow$ 
    L1 is P-1, L2 is P-2, L3 is P-3, L4 is P-4,
    R1 is P+1, R2 is P+2, R3 is P+3, R4 is P+4.
  
```

In iteration 1 non recursive rules like the following are induced:

```

tag(A, B, adj)  $\leftarrow$  word(A, B, portuguesa), !.
tag(A, B, n)  $\leftarrow$  word(A, B, documento), !.
  
```

These rules are defined solely in terms of the background predicates `word/3`. They do not depend on the context of the word to be tagged. Before proceeding to iteration 2 we add these rules to the background knowledge.

In iteration 2, some words can be tagged using the rules induced in iteration 1. Now these rules are defined in terms of the word to tag and the context. In this second iteration we also find many non recursive rules. In subsequent iterations more clauses will appear until the stopping criterion is satisfied.

Therefore recursive rules like the following appear:

```

tag(A, B, art)  $\leftarrow$ 
    window(A, B, L1, L2, L3, L4, R1, R2, R3, R4), tag(A, L1, prep),
    tag(A, R1, n), tag(A, L2, n), tag(A, R2, virg), tag(A, L3, prep), !.
tag(A, B, art)  $\leftarrow$ 
    window(A, B, L1, L2, L3, L4, R1, R2, R3, R4), word(A, B, a),
    tag(A, R2, prep), tag(A, R3, n), tag(A, R4, prep), !.
  
```

In general, the total number of iterations depends on the data, the language, and the underlying learning algorithm employed. For the experiments described in this article, the typical number of iterations was 5.

## 4 The Case-Based Approach in the Iterative Strategy

By observing the partial results of each theory  $T_1, T_2, \dots$  produced by iterative induction it was clear that the last theory in the sequence is responsible for a large number of wrong answers. This is not surprising, since the examples left for the last iteration are the most difficult ones. Previous iterations failed to find good clauses in the given language. To improve the results we have shifted the bias in the last iteration by applying case based reasoning.

We have two main approaches to compute similarity between cases, syntactic and semantic (Aamodt 1994). In the syntactic methods the similarity is inversely proportional to the distance between cases, and a case is described as a vector of feature-values and a corresponding class. On the other hand, semantic methods employ background knowledge and are able to explain cases and use these explanations to retrieve and adapt cases. For this work, we adopted this second view of cases.

Preliminary experiments we have conducted employed syntactic methods. In these experiments we defined the cases as set of features corresponding to a neighborhood of the word to tag of length 11. The overlapping metric used divided the number of matching features by the total number of features representing the context of the word. Weights of features were manually set. The best result was not better than previous results obtained with rules only (Table 1 and Figure 1). Besides, setting the appropriate weight to each position in the window is a difficult task. Although closer neighbors tend to be more relevant for tagging, more distant words may be important in certain contexts.

These results motivated the use of the semantic approach to CBR. For that, we developed the new algorithm RC2 (Rules and Cases) that constructs explanations from cases, and uses explanations to classify new cases. Explanations are constructed in different levels of generality, enabling different views of the case. These different views correspond to different case filters that are suited for a particular kind of cases. A consequence of this is that, differently from usual case-based systems, we do not use a fixed metric to retrieve cases, but an appropriate set of conditions according to the new case being analyzed. In the following sections we describe in detail our concept of cases and the construction and use of explanations.

## 5 Cases

To decide which tag  $T$  should be assigned to a given word, in a given position  $P$  in a sequence, the new case description must take into account either the word at that position or the context of that word or both. For practical reasons, this context was limited to the tags on the five positions at the left of  $P$  and five positions at the right of

P (a window of size 11). In general, the context may include any information regarding that word. In our approach, case descriptions can be regarded as *ground* clauses of the form:

```
tag(S,P,T)←
  window(P,L1,L2,L3,L4,L5,R1,R2,R3,R4,R5), word(S,P,W),
  tag(S,R1,TR1), tag(S,R2,TR2), tag(S,R3,TR3), tag(S,R4,TR4),
  tag(S,R5,TR5), tag(S,L1,TL1), tag(S,L2,TL2), tag(S,L3,TL3),
  tag(S,L4,TL4), tag(S,L5,TL5).
```

For example, the case associated to the position 2 in sentence s1 is described by the following ground clause:

```
tag(s1,2,n)←
  window(2,1,0,-1,-2,-3,3,4,5,6,7), word(s1,2,car),
  tag(s1,3,v), tag(s1,4,adj), tag(s1,5,dot), tag(s1,6,pr),
  tag(s1,7,v), tag(s1,1,art), tag(s1,0,'?'), tag(s1,-1,'?'),
  tag(s1,-2,'?'), tag(s1,-3,'?').
```

Notice that the context here corresponds to the literals that define the neighborhood of the position being classified. Also notice that a case corresponds to a maximally specific clause in terms of the description of its context.

## 6 Case Explanations

For each case we have a set of explanations. These are clauses that are more general than the case, given some language constraints.

Let C be a case and L be a clause language, the set of explanations  $exp(C)$  is

$$exp(C) = \{ E: A \rightarrow B \in L \mid E \text{ } \theta\text{-subsumes } C \}$$

As described below we will construct a subset of these explanations and select only the ones applied to a large number of cases. That will be measured by the support and confidence parameters defined as:

$$Support(A \rightarrow B) = \# \{ \text{true instances of } A \wedge B \}, \text{ and}$$

$$Cf(A \rightarrow B) = \# \{ \text{true instances of } A \wedge B \} / \# \{ \text{true instances of } A \}.$$

One explanation associated to the case in the previous section could be

```
tag(S,Pos,n)←
  window(Pos,L1,L2,L3,L4,L5,R1,R2,R3,R4,R5), word(S,P,car),
  tag(S,R1,v), tag(S,R2,adj), tag(S,R3,dot), tag(S,R4,pr),
  tag(S,R5,v), tag(S,L1,art), tag(S,L2,'?'), tag(S,L3,'?'),
  tag(S,L4,'?'), tag(S,L5,'?').
```

Other explanations can be obtained by deleting literals in the body of the clause.

Each explanation built by RC2 is obtained by generalizing each pair of cases of the same class. This is possible since we are dealing with a relatively small set of residual cases (about 400). The number of explanations can also be controlled by defining appropriate support and language bias. To obtain the generalization of two cases  $C_1$  and  $C_2$ , we first compute the *least general generalization* (lgg) of  $C_1$  and  $C_2$  and then remove literals of the form  $tag(X,Y,Z)$  where  $Y$  or  $Z$  are variables that occur nowhere

else in the clause. The explanations with support and confidence above given thresholds are stored in a base of explanations.

Besides the support and confidence, each explanation is characterized by its level of generality. This is the number of literals defining the context used in the explanation.

---

**Algorithm 2:** Explanation Base Construction

---

**Given**

Cases  $C$ ,  
 Background knowledge  $BK$ ,  
 Minimal support  $MS$ , minimal confidence  $MC$

**Do**

For each pair of cases  $(c_1, c_2)$  in  $C$ , with the same class  
     construct explanation  $exp = \text{filtered lgg}(c_1, c_2)$   
     such that  $Support(exp) \geq MS$ ,  
                      $Cf(exp) \geq MC$

---

We call this set of explanations Explanation-Base.

---

## 7 Using Explanations

The tagging of a corpus using a theory produced by iterative induction is also done in an iterative way. Initially, no word occurrence in the corpus is tagged. Then, the induced theories  $T_1, T_2, \dots, T_n$ , are applied in sequence. Each theory tags some of the words, and uses the tagging done by previous theories. In the last iteration we use the case based classification.

To tag one occurrence of a word in a sentence using a explanations, we first represent that occurrence as a case in the case language defined (section 5). As described there, the case contains the context information for that particular position in the sentence. Since many of the words have already been tagged by previous iterations, the context of one word contains the known tags neighboring that word. After this, we look for the explanation in the explanation-base that maximizes the similarity measure described below. This is, in some aspects, similar to a regular instance based approach. The main difference is that here we may have, in the explanation-base, a set of explanations with different levels of generality and different views for each training case.

Given a case explanation  $A$  and a case  $B$ , the similarity metric used combines an overlapping metric, given by the number of matched literals divided by the total number of literals in the case explanation ( $Dist$ ), with the confidence of the explanation used ( $Cf$ ) and the level of generality of the explanation ( $N$ ).

$$Sim(A, B) = Dist \times Cf \times \log(N \times 10/M)$$

Where  $M$  is the number of literals in the maximally specific explanation. The value of  $Sim$  ranges from 0 to 1. To count the matching literals of a case and an explanation, we first unify the head of the clause representing the case with the head of the clause

representing the explanation. One literal  $L_c$  in the body of the case matches with one literal  $L_e$  in the body of the explanation if they are the same. We are assuming that all the variables in the explanation will be instantiated after unifying its head.

When more than one explanation with the same level of generality and the same number of matched conditions apply to one case, it is preferred the explanation with higher confidence. The factor  $\log(N \times 10/M)$  is an *ad hoc* generality measure that gives more weight to more specific explanations. Considering that the approach is case-based, it is natural to prefer, under similar circumstances, explanations closer to the new case (the most specific ones). Experiments not reported here have confirmed that retrieving explanations by using this generality measure works better than structuring the retrieval process by level of generalization, starting with the most specific explanations.

A maximally specific explanation can be seen as the case itself. In this case, the confidence is typically 1,  $\log(N \times 10/M)$  becomes 1, and the similarity metric is reduced to the usual overlapping metric.

Note that  $\log(N \times 10/M)$  is negative when  $N < M/10$ . This happens when the explanation used has less than 10% of the literals of the most specific one. For the part-of-speech tagging approach described here, this generality measure ranged from 0 ( $N = 1, M = 10$ ) to 1 ( $N = 10, M = 10$ ).

It is important to note that the main difference between an explanation and a rule lies in the fact that when using one explanation we do not have to match all literals. Besides, in a rule-based approach it is necessary to select, from all the hypotheses, an appropriate set of rules. Here we only have to store the explanations.

## 8 Results

In the experiments conducted, we observed that the use of a case-based approach in the last iteration of an iterative induction strategy instead of induced rules improves the accuracy results.

We have used the described approach in a set of experiments with a corpus in Portuguese text containing more than 5000 words. The corpus had been manually tagged.

The corpus was divided into a training set with the first 120 sentences (about 4000 words), and a test set with the remaining 30 sentences (1000 words). The theories were induced using the information in the training set only, and then we measured the success rate of the theories on the test sets. Notice that the learning task we consider here starts with *no dictionary*. In fact, the dictionary is learned and is expressed as rules that will be part of the final theory produced. In this experimental framework, tagging words of the test set is a hard task since approximately 30% of the words do not occur in the training set.

We now give some details about the synthesis of the theory associated with the result shown in Table 1. In the first four iterations a large number (more than 350) of rules are induced. Some 350 appear in iteration 1 and do not take the context into account. In the experiments, the minimal confidence of the rules for each iteration was 0.8. The minimal support was 2. In iteration 2 many recursive rules (about 200)



appeared. The iterative induction algorithm went through three more iterations. The number of rules induced at each iteration tends to decrease very rapidly.

**Table 1.** Success rates over the test with Lusa corpus

| Algorithm                            | Ac. Test |
|--------------------------------------|----------|
| Iterative CSC(RC1) with CBR at it. 5 | 0.792    |
| Iterative CSC(RC1) with rules only.  | 0.806    |
| Iterative CSC(RC1) with RC2 at it. 5 | 0.836    |

Table 1 shows the overall success rates obtained by using iterative induction with each one of three different algorithms in the last iteration (it. 5).

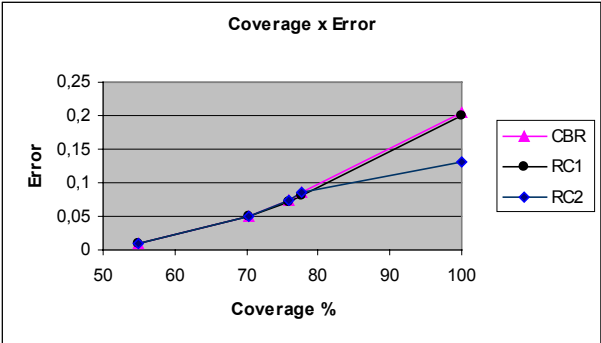


**Fig. 1.** Coverage  $\times$  Error. The first four iterations use the CSC(RC1) algorithm and in the last one we use the algorithms CBR, RC1, and RC2

Figure 1 shows the coverage vs. error rate obtained using the CSC(ALG) in each iteration. In the first 4 iterations ALG is the rule learner RC1. In the last iteration we have used the algorithm CBR (Case Based Reasoning with an overlapping metric), RC1 with a new set of parameters of quality to answer the remaining cases, and RC2 (using explanations).

The total number of iterations of the learning process depends on the data, the language, and the quality parameters (minimal confidence, support and selection algorithm). The inductive induction stops when the coverage in a given iteration is close to zero. This strategy yielded 5 iterations.

In the case of RC2, the untagged words at iteration 5 (about 400) were stored in a case-base and used to construct explanations (about 1300). The result shown for CBR in Table 1 was the best one achieved using a simple overlapping metric, and setting manually the weights.

## 9 Related Work

The system SKILit (Jorge & Brazdil 1996, Jorge 1998) used the technique of iterative induction to synthesize recursive logic programs from sparse sets of examples.

Many other ILP (Inductive Logic Programming) approaches to the task of part-of-speech tagging exist. The ones that are more directly related to our work are (Cussens

1997) and (Dehaspe 1997), where relational learning algorithms are employed in the induction of rule based taggers. More recently, Cussens et al. (1999) used the ILP system P-Progol to tag Slovene words. Lindberg and Eineborg (1999) used P-Progol to induce constraint grammars for tagging of Swedish words. And using linguistic background knowledge. Horv  th et al. (1999) tried different learning algorithms for tagging of Hungarian. One of the systems that obtained good results was RIBL, a relational instance based learning system.

In (Liu et al. 1998) a propositional learning algorithm was proposed that is similar in structure to CSC. The main differences are that CSC is relational and it is used here in an iterative way.

The methodology proposed here is one of a number of possible hybrid approaches combining cases and rules. The main motivations found in the literature for this combination are efficiency improvement and accuracy improvement. For example, Golding and Rosenbloom (1996), use a set of approximately correct rules to obtain a preliminary answer for a given problem. Cases are used to handle exceptions to the rules. Rules are also used for case retrieval and case adaptation. This approach yielded good accuracy results in the task of name pronunciation. Domingos (1996) proposes a propositional framework (and the learning system RISE) for the unification of cases and rules by viewing cases as most specific rules. The class of a new example is given by the nearest rule (or case) according to a given distance function. Rules are constructed by generalizing examples and other rules. Only generalizations that improve global accuracy are maintained. Our approach differs from RISE in some aspects. First, ours is a relational approach that can use background knowledge. Second, contrary to what happens in RISE, when an explanation is generated it does not replace the cases being generalized. We believe that this use of redundancy is important for a difficult set of examples like the ones treated in the last iteration of the inductive process. Another difference is that we use rules (in the first iterations) while these have a satisfactory quality, and cases only when the rule language exhausts.

## 10 Conclusion

In an inductive process such as iterative strategy, where the most visible patterns (represented as rules) are identified first, we typically get to a set of residual examples that cannot be reliably captured by the initial bias. As can be seen in Figure 1, the effectiveness (coverage) of the bias decreases from iteration to iteration.

The paradigm shift in the last iteration, when using a case based approach with explanations (RC2), improves significantly the accuracy in the iteration and overall. Case explanation was able to explore particularities of the cases not explored by the language bias in the previous rule-based inductive process.

Generating all explanations could be intractable for large corpora. However, the iterative approach used leaves only a relatively small set of cases for the last iteration.

The methodology proposed here has also explored and formalized some concepts such as case explanation, context, similarity assessment considering semantic aspects, as well as the use of background knowledge to understanding and retrieve cases.

Although the iterative learning process is described here as starting from scratch, previously acquired tagging knowledge could have been used before learning. Likewise we may have some words tagged before using the theories induced or the explanation-base constructed for tagging. Since we are using a first order setting, richer background knowledge can also be used in the learning process. However, this would probably motivate some more elaboration of the explanation matching concept.

## Acknowledgements

The authors would like to thank the support of project Sol-Eu-Net IST 1999 - 11495, project ECO under Praxis XXI, FEDER, and Programa de Financiamento Plurianual de Unidades de I&D. The first author would also like to thank the support of CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil. The Lusa corpus was kindly provided by Gabriel Pereira Lopes and his NLP group.

## References

1. Aamodt, E. Plaza Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, Vol. 7 Nr. 1, (1994), 39-59.
2. Cussens, J.; Dzeroski, S.; Erjavec, T.: Morphosyntatic Tagging of Slovene Using Progol. *Proceedings of the 9<sup>th</sup> Int. Workshop on Inductive Logic Programming (ILP-99)*. Dzeroski, S. and Flach, P. (Eds). LNAI 1634, 1999.
3. Cussens, J.: Part of Speech Tagging Using Progol. In *Inductive Logic Programming. Proceedings of the 7<sup>th</sup> Int. Workshop on Inductive Logic Programming (ILP-97)*. LNAI 1297, 1997.
4. Domingos, P.: Unifying Instance-Based and Rule-Based Induction. *Machine Learning* 24 (1996), 141-168.
5. Golding, A. R.; Rosenbloom, P.S.: Improving Accuracy by Providing Rule-based and Case-based Reasoning. *Artificial Intelligence* 87 (1996), 215-254.
6. Horváth, T.; Alexin, Z.; Gyimthy, T.; Wrobel, S.: Application of Different Learning Methods to Hungarian Part-of-Speech Tagging. *Proceedings of the 9<sup>th</sup> Int. Workshop on Inductive Logic Programming (ILP-99)*. Dzeroski, S. and Flach, P. (Eds). LNAI 1634, 1999.
7. Jorge, A.; Brazdil, P.: Architecture for Iterative Learning of Recursive Definitions. *Advances in Inductive Logic Programming*, De Raedt, L. (Ed.), IOS Press, 1996.
8. Jorge, A. Lopes, A.: Iterative Part-of-Speech Tagging. *Learning Language in Logic (LLL) Workshop*, Cussens, J. (Ed.), 1999.
9. Jorge, A.: *Iterative Induction of Logic Programs: an approach to logic program synthesis from incomplete specifications*. Ph.D. thesis. University of Porto, 1998.
10. Lindberg, N; Eineborg, M: Improving Part-of-Speech Disambiguation Rules by Adding Linguistic Knowledge. *Proceedings of the 9<sup>th</sup> Int. Workshop on Inductive Logic Programming (ILP-99)*. Dzeroski, S. and Flach, P. (Eds). LNAI 1634, 1999.
11. Liu, B.; Hsu, W.; Ma, Y.: Integrating Classification and Association Rule Mining. In *Proceedings of KDD 1998*: pp. 80-86. 1998.

# An Architecture for Knowledge Intensive CBR Systems<sup>\*</sup>

Belén Díaz-Agudo and Pedro A. González-Calero

Dep. Sistemas Informáticos y Programación  
Universidad Complutense de Madrid, Spain  
{belend, pedro}@sip.ucm.es

**Abstract.** In this paper we describe a domain independent architecture to help in the design of knowledge intensive CBR systems. It is based on the knowledge incorporation from a library of application-independent ontologies and the use of an ontology with the common CBR terminology that guides the case representation and allows the description of flexible, generic and homogeneous CBR processes based on classification.

## 1 Introduction

Any knowledge-based system (KBS) achieves its reasoning power through the explicit representation and use of different kinds of knowledge about a certain domain. Although in a CBR system the main source of knowledge is the set of previous experiences, our approach to CBR is towards integrated applications that combine case specific knowledge with models of general domain knowledge, mainly about the domain terminology. The more knowledge is embedded into the system, the more effective is expected to be. The major problem associated with this knowledge intensive CBR approach is the so called knowledge acquisition bottleneck (common for every KBS).

We make an issue of domain knowledge acquisition and study how the ontological engineering community efforts could help us to acquire the knowledge needed in a knowledge intensive CBR application. The goal of our current work is to formalize this approach to CBR and provide a tool and a methodology to assist during the design phase of CBR application development. Our main contribution is the definition of a domain-independent architecture to help in the integration of ontologies for CBR applications. The core of this architecture is CBROnto, an ontology incorporating the common CBR terminology that is used to guide the domain ontologies integration, and that will be the base of a future system to support the design of knowledge intensive CBR applications. This paper discusses the issues involved, but the whole system implementation is far from its final form.

Section 2 introduces the ontological reuse we propose for the CBR knowledge acquisition process and Section 3 describes CBROnto, the CBR ontology we have

---

<sup>\*</sup> Supported by the Spanish Committee of Science & Technology (CICYT TIC98-0733)

developed. In Section 4 we sketch the main ideas of the CBR processes defined within our architecture. Finally the conclusions, advantages and shortcomings of the current framework are discussed.

## 2 The Synergy of the Ontological and CBR Communities

The word *ontology* is used as a technical term by different groups to mean slightly different things. The more well-known definition for *ontology* is: “An ontology is a specification of a conceptualization” [8]. To clarify, an ontology:

- Expresses the consensus knowledge of a community of people.
- Defines the basic terms and relations comprising the vocabulary of a topic area, and contains precisely defined terms that can be used to describe and understand more complex descriptions.
- Can be reused and serve as a starting point to construct different knowledge-based applications.

Most of the KBSs (including the KB CBR systems) have some reusable ontological content but it is often influenced by the specific task, the restrictions of the representation language, and the specific inference procedures employed.

We state that ontologies can be useful for designing knowledge intensive CBR applications because they allow the knowledge engineer to use knowledge already acquired, conceptualised and implemented in a formal language, reducing considerably the knowledge acquisition bottleneck. Moreover, the reuse of ontologies from a library also benefits from their reliability and consistency [6].

We know of little interactions among the CBR community and the ontological community although the knowledge in an ontology is specially well-suited to be shared, and many CBR systems codify this kind of domain knowledge. Ontologies may help in the creation of complex, multirelational knowledge structures to support the CBR processes.

### 2.1 The Ontology Server

The Ontology Server (OS) (see [6] for a complete description and references) is a set of tools and services that support the building of shared ontologies between geographically distributed groups. It was developed in the context of the ARPA Knowledge Sharing Effort by the Knowledge System Laboratory at Stanford University. This server is an extension of the language Ontolingua<sup>1</sup>. The ontology server architecture manages a library of ontologies to be reused and provides a HTML interface to build, modify and browse ontologies; lexical and syntactic analyzers to avoid incompleteness, inconsistencies and redundant knowledge; and a set of translators to various knowledge representation languages as CLIPS, CML RE, EPIKIT, LDL, KIF, LOOM, OKBC, PROLOG. The OS and Ontolingua have been accepted by the knowledge-sharing community as the main tool to implement ontologies mainly due to its complete set of translators.

<sup>1</sup> <http://www-ksl-svc.stanford.edu:5915/> or <http://granvia.dia.fi.upm.es:5915/>

## 2.2 Description Logics

Ontologies must be codified in a formal language. Description Logic based languages (DLs) are commonly used to implement ontologies, and it is the technology we use in our model to formalize aspects of representation and reasoning. DLs, rooted in the KL-ONE family [3] and the frame systems, are characterized by its expressiveness and clearly defined semantics. The implementation language we use is LOOM [9], one of the destination languages of the OS translators.

DLs capture the meaning of the data by concentrating on entities (grouped into classes or concepts) related by relationships. This intuition is shared by formalisms such as semantic data models, semantic networks or frame systems. More important than the DLs representational characteristics are its reasoning mechanisms. The most important characteristic is the checking of incoherencies and the organization of the concepts on a taxonomy that the system automatically builds from the concept definitions. This is possible because of the clear and precise semantic of concept definitions that avoid the user to put the concepts in the correct place of the hierarchy (as is the case in frame systems, which provide inheritance but not classification).

DLs reasoning mechanisms and deductive inferences are based on *subsumption* and *instance recognition*. Subsumption determines if a term is or not more general than another, and instance recognition finds all the concepts that an individual satisfies. Furthermore, completion mechanisms perform logical consequences like inheritance, combination of restrictions, restriction propagation, contradiction detection, and incoherent term detection. These mechanisms will be used during the ontology integration, the case representation and, in general, as the base for all the CBR processes.

## 2.3 How the Ontologies Are Used

We are interested in three types of ontologies [6] from those provided by the OS: (1) the *domain ontologies* provide the vocabulary for describing a domain and interpreting a description of a problem in that domain; (2) the *task ontologies* provide the vocabulary for describing terms involved in the problem-solving processes, which could be attached to similar tasks that may, or may not, be in the same domain; and (3) the *common sense ontologies* include a wide-range amount of foundational knowledge as time, space, or causality.

The activities performed by the CBR application designer to model a domain, and to formalize it as LOOM knowledge base, are summed up as follows.

1. The designer begins with a preliminary idea of what domain is to be modelled, and selects from the library those ontologies that are potentially useful. For example, if we were modelling the *used-car* domain, a sensible choice would be the VEHICLES ontology which comprises knowledge about “*vehicles which are typically bought and sold through the classified ads ...*”<sup>2</sup>

---

<sup>2</sup> Excerpt from the OS documentation.

Our tool may suggest the designer some other ontologies depending on the ontologies previously chosen. In the example, the incorporation of the `VEHICLES` ontology will suggest the `PRODUCT-ONTOLOGY` inclusion, which “*defines the terms used for describing products, objects that are typically bought and sold . . .*”<sup>2</sup> This, in its turn, will lead the system to consider the terminology within the `SCALAR-QUANTITIES` and `STANDARD-UNITS` ontologies.

2. The domain terminology from the ontologies has to be integrated as two term hierarchies: the concept hierarchy rooted by the `THING` concept, and the relation hierarchy, rooted by the `BINARY-TUPLE` relation. The designer chooses where, within those hierarchies, the ontology components have to be placed. The system, in its turn, combines and propagates through inheritance the restrictions included in the terms, and eventually signals contradictions and incoherencies. Then, it is again the designer who must solve the integration problems detected. Anyway, we must point out that the issue of coherent integration of definitions from different ontologies is still an open problem [13].
3. Due to the fact that ontologies are very general and reusable, sometimes all the definitions inside an ontology are not useful for our concrete domain model. The elimination of not relevant terms is not essential but, in our approach, will effect on the final system efficiency and quality because the search space will be smaller and contain only relevant terms. Notice that the selection of a definition from an ontology can provoke the automatic inclusion of others interrelated definitions that can not be eliminated and conversely, the elimination of some definitions could cause others to be erased.
4. Mechanisms are also provided to allow the inclusion of new definitions, just in case some useful specific definitions for our domain are not included in the chosen ontologies. Unfortunately, when there are not appropriate ontologies to be reused, an effort is needed to build a new ontology (or knowledge base).

In its final form, our tool will allow the term hierarchies graphic visualization (the OS doesn’t provide with this functionality) and select, merge, eliminate, move, add or modify their definitions.

### 3 The CBR Ontology

As it was described in Section 2, our approach proposes the use of an ontology library to build the domain model for knowledge-rich CBR applications. To take advantage of this domain knowledge, the CBR knowledge needed by the processes, or at least part of it, should be expressed in a similar way. We have developed an ontology for CBR (`CBROnto`) that provides the vocabulary for describing the elements involved in the CBR processes. `CBROnto` serves two purposes: the integration between the domain ontologies and the CBR process knowledge; and as a domain-independent framework to design CBR applications.

With this approach, the designer of a knowledge rich CBR application does not only borrow domain terminology from the ontology library but also CBR

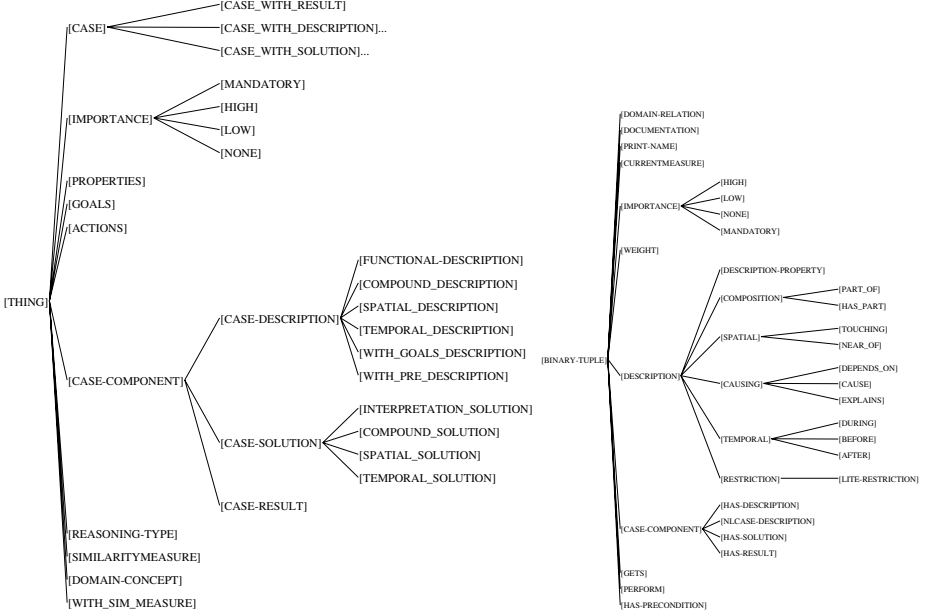


**Fig. 1.** The process support knowledge

terminology from CBRonto. CBRonto seeks to capture semantically *important* terms and the representation primitives commonly used in the case-based representation languages. It should be categorized inside the *task ontologies* because it provides a vocabulary for describing terms involved in the problem-solving CBR processes. Figure 1 shows a fragment of the CBRonto hierarchies.

CBRonto reveals our current view of certain CBR dependent but domain-independent terms that make possible different types of CBR, and that are used as the junction between the domain knowledge and the processes we define with a domain-independent perspective (see Figure 2). After the domain modelling, the phase of integration is based on classifying the domain terms with respect to the CBRonto terms. That mechanism allows the CBR processes being domain independent because they only refer to the CBRonto terms, that are correspondingly linked to the domain terminology by the classification mechanism.

Next subsections describe the main characteristics of CBRonto, the place it takes during the case representation, and how it is used to integrate the domain knowledge to be utilized by domain-independent CBR processes.

### 3.1 Case Representation

The cases in the case base should be described somehow by mean of the vocabulary provided by the domain model. The issue of case representation involves deciding the type and the structure of the domain knowledge within the cases. Efficiency pushes many CBR systems to use simple case representations that typ-



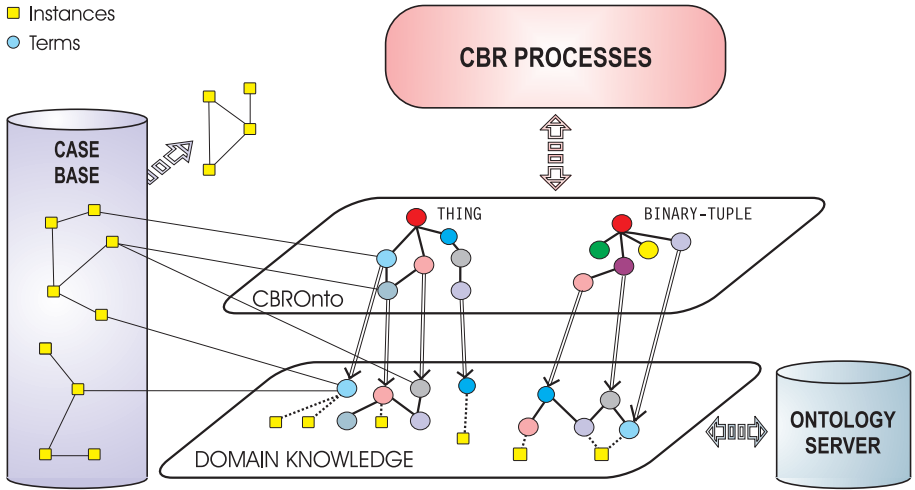


**Fig. 2.** CBROnto as a join between the domain terminology and the CBR processes

ically contain two sets of attributes, problem and solution features, and where there are no relationships or constraints between the features of a case. That is not our choice. We don't want to restrict the cases to be monolithic units of a fixed format because our framework is intended to be suitable for different types of CBR [4]. Our aim is to propose a rich framework to represent cases based on the terminology from the CBROnto together with a reasoning system that works with such representations.

**The Case Representation Language** In the CBROnto origins, the first decision was the definition of a *primitive* concept **CASE**. We will call *case-type* concepts to the **CASE** subconcepts. That way, cases can be represented as instances of the case-type concepts and will be described by using both the domain vocabulary provided by the domain model, and the CBR vocabulary provided by the CBROnto. Cases are represented as instances of different **CASE** subconcepts, so they won't have, in general, the same structure. Besides, the concrete cases (**CASE** instances) may add other proper features to the fixed structure inherited through the case-type concepts.

The designer will define case-type concepts to represent the new types of cases. The CBROnto vocabulary is used to guide the definition of these concepts by providing with CBR semantically important terms as *has-description*, *has-solution*, *has-result*, *similarityMeasure*, *weight*, *goal*, *precondition*, or *description-property*.

Our case representation language is based on the LOOM instance definition language and on the CBROnto terminology. This representational framework allows complex structures and does not restrict the possible relations among the parts of a case, facilitates the definition of cases having different structures, is

able to handle incomplete cases and allows default values (by inheritance). The case instances may be related with other individuals, and in particular with other case instances, i.e. a case can be related with cases that are cases themselves.

We propose the use of an instance of the concept **CASE-DESCRIPTION** to represent the description of a case. The **has-description** relation links a **CASE** instance with the individual representing the description of this case. CBR<sub>Onto</sub> includes the following components to describe a **CASE-DESCRIPTION** instance. The use of different components builds different types of case descriptions.

- The kind of reasoning the case will be used for. We use (by now) the following: **diagnosis**, **evaluate**, **explain**, **design**, **solve**, and **search**. They are represented as instances that will be linked to the case description instance by means of the **kind\_of\_reasoning** relation.
- The goals achieved by the case. The goals will be specific for the concrete domain and will be represented as instances of the CBR<sub>Onto</sub> **GOAL** concept.
- The restrictions to be considered before applying the solution.
- Other suitable properties to describe the case: **composition**, **causing**, **temporal**, or **description-property** relations. (see Figure 1 (right)).

That way, the CBR processes can take advantage of this explicit definition of certain parts of the case structures, through the concept and relation hierarchies and the DLs inference mechanisms.

The solution of a case is represented as an instance of the **CASE-SOLUTION** concept. The **has-solution** relation links a **CASE** instance with the **CASE-SOLUTION** individual representing its solution. We use a general perspective, because the solution of a case depends very much on the kind of reasoning the case represents. The kind of solution is represented by classifying the solution individual below the **CASE-SOLUTION** subconcepts. It can be a designed component, a layout, a plan, or a diagnostic or interpretation for the current situation. As an example of the CBR<sub>Onto</sub> terms that might be used to describe a **CASE-SOLUTION** instance we cite the *spatial*, *temporal*, *composition* or *causing* relations to represent the temporal sequence of reasoning steps used to solve the problem represented by the case; the spatial layout of the pieces used to design a component, or the adaptation (dependency) knowledge used to build this solution and links to the cases used to make it. To finish with the case main parts, the result of a case might include components as the success or failure of the case, the explanation of a failure, or links to other possible solutions.

### 3.2 The Domain and CBR<sub>Onto</sub> Integration

As we have introduced, after the domain modelling phase, there is an integration phase where the CBR application designer relates the specific domain knowledge with the CBR<sub>Onto</sub> terms. This section aims to explain the basic mechanisms used to integrate the domain and the CBR<sub>Onto</sub> term hierarchies.

We use DLs classification to relate the specific domain terms with the CBR<sub>Onto</sub> terms. Suppose a domain relation that is used to describe a property of the

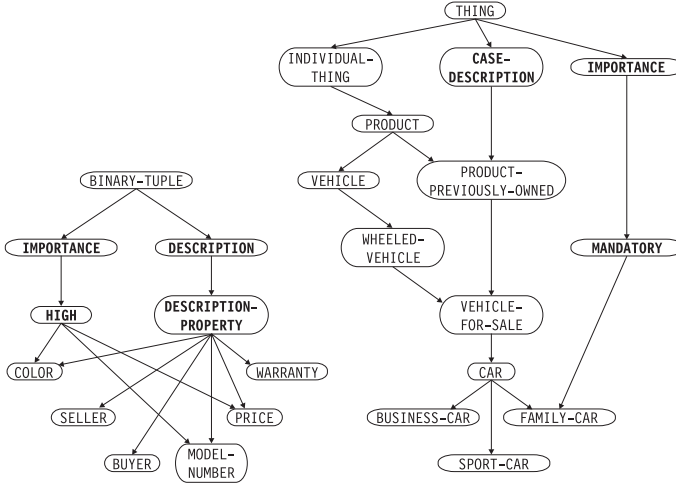


**Fig. 3.** Integration mechanisms based on classification (CBROnto terms in bold)

domain cases. For example, the `color` relation. With our framework, it will be classified as a subrelation of `description-property` because is a relation used to describe a domain property (see Figure 3). The same mechanism is used to classify other kinds of relations as `temporal`, `composition` or `spatial`.

We are using relation classification here, but the mechanism is similar in the concept hierarchy. For example, once we have modelled the used-car domain by means of the `VEHICLES` and `PRODUCT` Ontologies we want to represent the different types of cases. We would like to have cases representing second-hand products and without solution. Each case can include one or more products. We are building the case-type concept `CASE-PRODUCT-CLASS` with the structure of Figure 4 (left). The integration mechanism classifies the `PRODUCT-PREVIOUSLY-OWNED` domain concept below the CBROnto `CASE-DESCRIPTION` concept. With this representation the `PRODUCT-PREVIOUSLY-OWNED` instances (from the `Product` ontology) are used as the description components of the `CASE-PRODUCT-CLASS` cases. Due to the classification mechanism, instances of `FAMILY-CAR`, `SPORT-CAR`, `BUSINESS-CAR`, `CAR`, and `VEHICLE-FOR-SALE`, are also appropriate instances to be used to describe a `CASE-PRODUCT-CLASS` case.

Also based on classification, our framework provides with a way to express preferences between the terms. This mechanism can be used for many purposes, and either by the designer, the final user of the designed CBR application, or by the organization processes. The importance for the case descriptors used during retrieval can be expressed by classifying them under the `IMPORTANCE` terms: `MANDATORY`, `HIGH`, `LOW`, and `NONE`. The domain independent CBR processes will prefer the domain relations classified under the `HIGH` relation and avoid the `NONE` classified ones. Figure 3 illustrates the use of the `HIGH` relation to strengthen the `color`, `model-number` and `price` domain relations; and the use of the `MANDATORY`

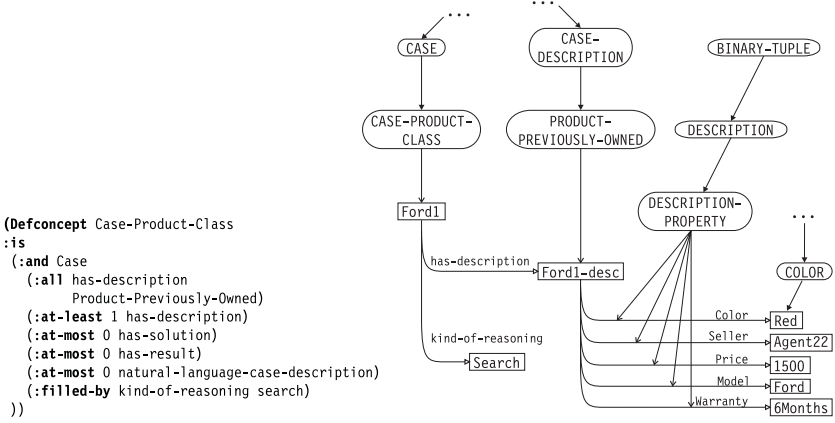


Fig. 4. Case definition example

concept to indicate that only the FAMILY-CAR type of CASE-DESCRIPTION should be considered for this retrieval.

The designer doesn't handle this low-level classification mechanisms nor classify one by one every domain term. Due to the inheritance mechanism only the top level terms in the hierarchies should be classified. Besides, we are developing a graphical environment to help this integration between the domain and the CBROnto terminology.

## 4 The CBR Processes

The described representational framework facilitates general and homogeneous CBR processes that refer to the CBR terminology and not to the specific domain terminology (see Figure 2). In this sense, the CBR processes are domain-independent but they are guided by the domain terminology organized below (in the subsumption hierarchies) the CBROnto terms. We are working nowadays in the development of the CBR processes and we are not elaborating here their details but only enumerating some of the alternatives we are considering.

Several alternatives can be chosen in our system to *index* the cases. The straight one is the use of the domain terminology as the case organization structure. That's the approach we used in [5]. Other approach [14] is let the designer explicitly define indexes as new DLs concepts. The links between indexes and between cases and indexes are automatically (semantically) computed. The alternative we are mainly using is the computation of a different index structure by inductive techniques guided by the domain knowledge.

With regard to *retrieval* and *similarity* assessment, the straight possibility [1] is the use of the LOOM query language to enable the user to describe the current situation and interests. Also, in the line of [12,14] a similarity term (concept) could be explicitly computed (and automatically classified) to represent in a declarative way the similarity and differences between the cases, expressed with

```

System question:    GIVE A NAME FOR THE NEW CASE
User answer:        Ford1
System question:    CLASSIFY THE NEW CASE CHOOSING FROM THE LIST.
System internal:    (Get-subconcepts CASE)   Response: Case-Product-Class
User answer:        Case-Product-Class
System question:    HOW DO YOU DESCRIBE YOUR CASE?
System internal:    (Get-subconcepts PRODUCT-PREVIOUSLY-OWNED)
; The concept PRODUCT-PREVIOUSLY-OWNED is used due to the restriction
;(:all has-description PRODUCT-PREVIOUSLY-OWNED) in the case-product-class concept
Response:          PRODUCT-PREVIOUSLY-OWNED, VEHICLE-FOR-SALE, CAR,
                    FAMILY-CAR, SPORT-CAR, BUSINESS-CAR.
User answer:        FAMILY-CAR
System internal:    (createm 'Ford1-desc FAMILY-CAR)
                    (tell (:about Ford1 Case-Product-Class
                           (:all has-description FAMILY-CAR)
                           (has-description Ford1-desc)))

Response:          Recognition changes in the KNOWLEDGE BASE:
                    entry: Ford1-desc CASE-DESCRIPTION
                    entry: Ford1 CASE-PRODUCT-CLASS
                    entry: Ford1 CASE_WITH_DESCRIPTION
                    entry: Ford1 CASE
System internal:    (get-subrelations DESCRIPTION)
Response:          COLOR, SELLER, BUYER, MODEL, PRICE, WARRANTY
System question:    WHAT'S THE COLOR OF THE CASE?
System internal:    (retrieve ?x (range color ?x))
Response:          |concept| (COLOR)
                    (get-instances (fc COLOR))
                    (GRAY BLUE BLACK RED GREEN WHITE YELLOW ORANGE)
User answer:        RED
System internal:    (tellm (:about Ford1 (COLOR Red))
.....

```

**Fig. 5.** User interaction simulation

the domain terminology. Another possibility is the representational approach that assigns similarity meaning to the path joining two individuals. We are using CBR<sub>Onto</sub> to define different similarity components depending on the used terms: the *structural similarity* will be computed based on the composition relations (part-of, has-part), the *semantics similarity* is due to all the concepts and relations describing the meaning of the case, the *contextual similarity* depends on the case context relations and the *adaptation similarity* will use the dependency knowledge.

#### 4.1 A Case Definition Example

This section exemplifies the process of incorporating a new case to the case base. The used-car domain case structure represented by the **CASE-PRODUCT-CLASS** concept (see Section 3.1 and Figure 4 (left)) is very simple and doesn't illustrate all the representational possibilities, for example, cases with solutions and results, or more complex descriptions described by not simple properties. However, it exemplifies and facilitates the comprehension of the domain-independent and classification based mechanisms that provide access to the domain terminology through the CBR<sub>Onto</sub> terms.

Figure 5, simulates the user interaction to instantiate the CBR<sub>Onto</sub> concepts and relations. The key issue in the example is that the system questions are dynamically generated by querying the knowledge base with domain-independent questions referring only to the CBR<sub>Onto</sub> terms (*system internal* in the figure).

Notice that during the example, the domain terms are always reached because they are classified below the CBRonto terms.

The user chooses the **FAMILY-CAR** type of **CASE-DESCRIPTION**, and the system creates the individual **Ford1-desc**, which is an instance of the **FAMILY-CAR** concept; and an individual called **Ford1**, which is an instance of the concept **CASE-PRODUCT-CLASS** and that is related with **Ford1-desc** by the **has-description** relation. The meaning is that the **Ford1-desc** individual represents the description of the case **Ford1**. The next step is describing the **Ford1-desc** instance. The system will access the relation hierarchy and formulate a question for each **DESCRIPTION** subrelation. When possible, the system offers a set of fillers according to the range of each relation (as is the case with the **color** relation). Figure 4 (right) shows the **Ford1** case resultant from this user interaction.

## 5 Conclusions and Related Work

In this paper we have not aimed to describe all the terms within the CBRonto, mainly because we don't think it is complete but it is evolving with our current work. It only makes explicit certain CBR terms that are useful as a junction between the domain knowledge and the CBR processes defined with a domain-independent perspective. The use of domain ontologies provides a CBR application with the vocabulary for describing a domain and interpreting a description of a problem in that domain. The use of domain ontologies guides the construction of cases (and queries) and constitutes a warehouse of vocabulary to solve lexical, semantic and synonym problems. Besides, it avoids misunderstandings if cases are given by different sources and allows for a seamless integration of cases without requiring all the cases to have the same structure. As the main drawback of our approach we cite the ontology integration problem. When there are not appropriate ontologies to be reused, an effort is needed to build new ontologies or to integrate definitions from different ontologies [13]. Anyway, we consider this effort is not waste time if this knowledge is reused for other applications.

Although we aim to build a tool capable to fit many CBR approaches, we don't expect to contribute in the design of simple CBR applications, where cases can be attribute-value vectors, and where many optimised technologies exist. Our contribution is expected to be in the knowledge intensive applications where our system will allow for a quick way of design and prototyping a CBR application and study the results of the incorporation of certain domain knowledge.

Other works [1,10,14] use DLs to represent the cases and the domain knowledge for CBR systems, but none of them do it with a domain and application independent view as ours. As we did in [5,7], these works presume the existence of a DL knowledge base in the application domain, typically built ad hoc for a concrete application.

Other CBR systems codify specific domain ontologies. In [2] an ontology is developed for the representation of cases and adaptation knowledge for a CBR system that helps in the estimation of effort for software project. The ontology manages software project terms like **task**, **project**, **resource** and **deliverable**. Their

proposal fits in our architecture if the built project-effort ontology is included in the library of domain ontologies. More related with our CBR<sub>Onto</sub> is the Multis Ontology (see [6] for references), a task ontology that defines the terminology for scheduling. We intend to use it to enhance CBR<sub>Onto</sub> in the design of planning CBR applications. Also we plan to integrate the context ontology developed in [11] for its use in contextualized problem solving and learning.

The development of this system brings several lines of CBR future research. We will perform empirical studies comparing our system with other CBR shells, mainly for the CBR system design time, and the efficiency, quality of the results, suitability and effectiveness of the designed applications.

## References

1. Ashley K. & Alevén V., 1993: "A logical representation for relevance criteria", in *Topics in CBR* (Wess S., Althoff K. & Richter M., eds.), Springer-Verlag. 45, 47
2. Aarts R. J., 1998: "A CBR Architecture for Project Knowledge Management", in *Advances in CBR* (Smyth B. & Cunningham P., eds.), Springer-Verlag. 47
3. Brachman R. J., McGuinness D. L., Patel-Schneider P. F., Resnick L. A., & Borgida A., 1991: "Living with CLASSIC: When and How to Use a KL-ONE-Like Language". In *Principles of Semantic Networks*. Morgan Kaufmann Publishers. 39
4. Gebhardt F., VoB A., Gräther W., Schmidt-Belz B., 1997: *Reasoning with Complex Cases*. Kluwer Academic Publishers. 42
5. Gómez-Albarran M., González-Calero P. A., Díaz-Agudo B. & Fernández-Conde C., 1999: "Modelling the CBR Life Cycle Using Description Logics", in Procs. of the 3rd International Conference on Case-Based Reasoning (ICCBR'99). K.-D. Althoff, R. Bergmann & L. K. Branting (Eds.). 45, 47
6. Gómez-Pérez A., 1998: "Knowledge Sharing and Reuse". *The handbook on Applied Expert Systems*. By Liebowitz. ED CRC Press. 1998. 38, 39, 48
7. González-Calero P. A., Gómez-Albarran M., & Díaz-Agudo B., 1999: "Applying DLs for Retrieval in Case-Based Reasoning", in Procs. of the 1999 Description Logics Workshop (DL'99). 47
8. Gruber, T. "A translation Approach to portable ontology specifications". *Knowledge Acquisition*. Vol, 5. 1993. 38
9. Mac Gregor, R., 1991: "The evolving technology of classification-based knowledge representation systems", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge* (J. Sowa, ed.), 39
10. Napoli A., Lieber J., & Courien R., 1996: "Classification-Based Problem Solving in CBR", in *Advances in CBR* (Smith I. & Faltings B., eds.), Springer-Verlag. 47
11. Ozturk P. & A.Aamodt, 1998: "A Context Model for Knowledge-Intensive Case-Based Reasoning", *International Journal of Human-Computer Studies*. Vol.48,3. 48
12. Plaza E., 1995: "Cases as Terms: A feature term approach to the structured representation of cases". In Procs. ICCBR-95. 45
13. Pinto H. S., Gómez-Pérez A. & Martins J. P., 1999: "Some Issues on Ontology Integration", in IJCAI-99, Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. 40, 47
14. Salotti S. & Ventos V., 1998: "Study and Formalization of a CBR System using a Description Logic", in *Advances in CBR* (Smyth B. & Cunningham P., eds.), Springer-Verlag. 45, 47

# A Dynamic Approach to Reducing Dialog in On-Line Decision Guides

Michelle Doyle and Pádraig Cunningham

Department of Computer Science, Trinity College Dublin  
Dublin 2, Ireland.

{Michelle.Doyle, Pádraig.Cunningham}@cs.tcd.ie

**Abstract.** Online decision guides typically ask too many questions of the user, as they make no attempt to focus the questions. We describe some approaches to minimising the questions asked of a user in an online query situation. Questions are asked in an order that reflects their ability to narrow down the set of cases. Thus time to reach an answer is decreased. This has the dual benefit of taking some of the monotony out of online queries, and also of decreasing the amount of network request-response cycles. Most importantly, question order is decided at run time, and therefore adapts to the user. This approach is in the spirit of lazy learning with induction delayed to run-time, allowing adaptation to the emerging details of the situation. We evaluate a few different approaches to the question selection task, and compare the best approach (one based on ideas from retrieval in CBR) to a commercial online decision guide.

## 1 Introduction

Intelligent assistants on web sites, particularly e-commerce sites, are becoming increasingly common. Often these decision guides consist of a daunting list of questions that need to be answered in order to find the most suitable item<sup>1</sup>. Long-lived dialogs lead to monotony for the user, and do nothing to encourage user acceptance of such sites. Therefore an intelligent assistant which asks the *minimum* of questions in a dialog situation is desirable. This has the added benefit of minimising the number of network request-response cycles. A reduction in dialog length can be achieved through the production of *focused* questions.

Users often consult these kinds of systems with only a vague idea of their needs. Therefore a conventional one-shot retrieval strategy may not be ideal. Instead, a *multi-stage* retrieval process can be employed, in which the initial information is used to retrieve the first candidate set, and subsequent questions whittle down this subset until a manageable set of cases remains. At each step in the interaction with the user, we wish to ask them a question that will maximally discriminate between the

---

<sup>1</sup> See personallogic (<http://www.personallogic.com>) for example



candidate cases at that point. Thus the answers given by the user at each step influence the order of the questions. The incremental CBR technique (I-CBR) [3][14] employs such a retrieval strategy, and uses a simple information theoretic metric (*information gain*) to find the feature that best discriminates between the current set of retrieved cases. However decision guides mostly operate on data without class labels, such as product descriptions. The information gain measure used in incremental CBR *requires* class labels in order to decide the discrimination power of a particular feature. Therefore we must either find another metric to use with our multi-stage retrieval technique, or apply a clustering algorithm to our unlabelled data in a pre-processing step, allowing us to use the information gain measure.

Associated with this problem of *single* feature selection is that of feature *subset* selection. Feature subset selection techniques attempt to reduce the number of attributes used to describe an input in order to improve predictive accuracy. This is an important area of research in machine learning, as irrelevant attributes in the training data can lead to a decrease in the effectiveness of learning algorithms. A variety of feature subset selection techniques for supervised learning exist (see [1] for a discussion of sequential feature selection algorithms and further references). However, the absence of class labels makes unsupervised feature subset selection a more difficult problem, and there are fewer techniques to choose from. We investigated some of these techniques, and chose one whose evaluation function could be applied to our particular task. This function's performance was then compared to that of I-CBR's metric after clustering the same data.

In the following sections we discuss the issues further, describe our approaches to the problem, and present some results. In section 2 we discuss the I-CBR technique and the clustering algorithm we used, while our attempts to find a metric which works on unlabelled data are discussed in section 3. Results from the application of the metrics to a variety of case-bases are presented in section 4, where we also compare our approach (in terms of number of questions asked) to an existing online decision guide. We discuss our findings and possible uses of the technique in section 5.

## 2 Incremental Case-Based Reasoning and Clustering

### 2.1 Incremental Case-Based Reasoning

Incremental Case-Based Reasoning (I-CBR) is an incremental case-retrieval technique based on information-theoretic analysis [3][14]. The technique is *incremental* in the sense that it does not require the entire target case description to be available at the start, but in fact builds it up by asking focused questions of the user. The ordering of these questions reflects their power to discriminate effectively between the set of candidate cases at each step.

The technique was originally developed for use in the domain of fault diagnosis, where it is difficult to gather a complete case description in advance. In these types of problems, there is potentially a large amount of information that could be used to aid the diagnosis, but not all of this information is needed to solve the problem. Evidently, this approach need not solely be used on diagnosis problems. Indeed with a

decision guide, it is desirable to minimise the questions asked, and this approach can be used to do just that. As questions are asked in order of their discrimination power, the set of retrieved cases is whittled down at a maximal rate, consequently only a subset of the features should be needed to reach an answer.

The metric used in I-CBR, information gain, is presented again in [8]. After using this measure to decide the most discriminating feature for the current set of retrieved cases, the user is prompted for a value for this feature, the retrieved set is further reduced, and the most discriminating feature in relation to the new retrieved set is calculated. This continues until a manageable set of cases remains. In this way, the I-CBR approach differs from ID3, as no global decision tree is built; instead local, partial decision trees are built on the fly over a reduced set of cases.

Our implementation of I-CBR differs from the system described in [14]. We don't actually construct a local decision tree for retrieval; instead we use a spreading activation algorithm. The reason for this was to avoid the exact matching enforced by decision trees, and make use of the „fuzzy“ matching allowed by this approach.

At the first pass stage of retrieval a score is given to each case depending on its similarity to the target along the specified features. This score is produced through a „spreading activation“ process implemented using indexing structures. All those cases within a certain threshold of similarity are placed in the retrieved set. Fig. 1 shows this at work in a simple domain with only two features. The shaded area in Fig. 1(a) shows the retrieved set after the user provides the indicated value for feature f1. As each refining question is asked, the activation of each case in the retrieved set is updated according to their similarity to the evolving target description. The threshold of similarity is also updated, according to the total number of features entered. Those that are not within this threshold of similarity are excluded from the refined set of retrieved cases. Fig. 1(b) shows the refined set after determining the required value for feature f2. This process continues until the retrieved set is small enough. In the example shown, only feature f1 would have been asked if the goal were to retrieve 10 cases or less. Moreover, our system should also realise that f1 is more discriminating than f2, and therefore ask that feature first.

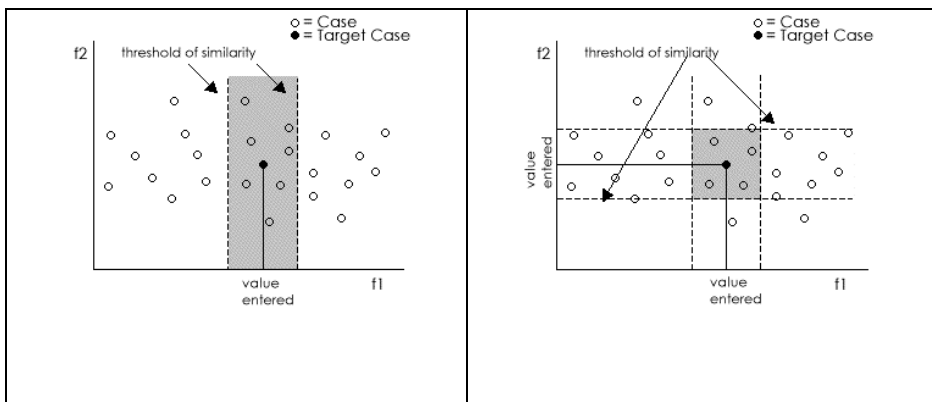


**Fig. 1.** (a) Shaded area is set of retrieved cases after user enters value for f1. (b) Shaded area is set of retrieved cases after user enters value for f2

We also implemented extensions to the basic algorithm for calculating information gain to deal with continuous features and unknown values in the data (see [13]).

As we will show later, using the information gain metric is an extremely good way of reducing the number of features needed to reach a consistent set of retrieved cases. However, it will not work for cases without inherent „classes“ or „outcomes“, as this information is needed to decide the discrimination power of a given feature. Therefore we experimented with pre-clustering the unlabelled data and then using the approach described above. The clustering algorithm used in these experiments is described below.

## 2.2 Clustering - The $k$ -Medoid Method

The clustering algorithm chosen was Kaufman and Rousseeuw's  $k$ -medoid algorithm [12]. Their approach was to attempt to find  $k$  *representative objects* (medoids) and to cluster the other objects according to their closeness to these. This has similarities to the well-known  $k$ -means algorithm with the important difference that the goal is to minimising the *average* dissimilarity of the representative object to all the other objects of the same cluster, whereas the  $k$ -means approach attempts to minimise the *average squared* difference. Therefore the  $k$ -medoid approach has the advantage of being less sensitive to outliers.

The optimal choice of  $k$  is a non-trivial problem and is crucial as a poor value can lead to unnatural clustering. Kaufman and Rousseeuw describe a method of graphically representing the cluster structure (a so-called *silhouette plot*), and suggest how a validity index, computed by this method, can be used to determine an appropriate choice of  $k$ . The basics of this approach were used in the system presented here (see [8] for details).

We have tested this clustering algorithm on a few data sets with known classes, and have had good results. For example, with the iris and thyroid<sup>2</sup> data sets, the accuracy scores of the clustering were 91% and 92% respectively.

## 3 Unsupervised Feature Subset Selection Methods

Feature subset selection can be viewed as a search through the space of possible descriptions of the training data [7]. The goal, in supervised learning, is to improve predictive accuracy through the elimination of redundant or misleading features. Similarly, in unsupervised learning the goal is to improve the efficiency of clustering and the quality of the clusters produced (irrelevant features can obscure cluster structure). There has been much research in the area of feature subset selection for supervised learning tasks, but the absence of class labels in unsupervised learning makes it a more difficult problem. However there has been some work in the area in recent years. We investigated two recent approaches and a feature extraction technique from the area of statistics. We compare them below in terms of their efficiency and the ease of using their evaluation functions with our incremental retrieval system.

---

<sup>2</sup> from the UCI ML Repository - <http://www.ics.uci.edu/~mllearn/MLRepository.html>

### 3.1 Feature Selection in Conceptual Clustering

The first unsupervised feature subset selection algorithm we investigated was Devaney and Ram's [7]. This was implemented in the area of conceptual clustering, using Fisher's [11] COBWEB system as their underlying concept learner. They use category utility as their evaluation function, which is already used by COBWEB to guide the process of creating concepts. This function measures the increase in the number of feature values that can be predicted correctly given a set of concepts, over those which can be predicted without using any concepts.

Their approach was to generate a set of feature subsets (using either Forward Sequential Selection (FSS) or Backward Sequential Selection (BSS), run COBWEB on each subset, and then evaluate each resulting concept hierarchy using the category utility metric on the first partition<sup>3</sup>. BSS starts with the full feature set and removes the least useful feature at each stage until utility stops improving. FSS starts with an empty feature set and adds the feature providing the greatest improvement in utility at each stage. At each stage the algorithm checks how many feature values can be predicted correctly by the partition – i.e. if the value of each feature  $f$  can be predicted for most of the classes  $C(k)$  in the partition, then the features used to produce this partition were informative or relevant. The highest scoring feature subset,  $B$ , is retained, and the next larger (or smaller) subset is generated using  $B$  as a starting point. The process continues until no higher category utility score can be achieved.

Since we are interested in feature ranking, not subset selection, this approach cannot be directly applied to our task. We need to be able to rate the importance of a *single feature* rather than a subset. However running COBWEB on just one feature would simply result in a biased concept hierarchy, and the category utility metric wouldn't tell us much in that situation. A better approach would be to first run the feature subset selection algorithm offline to get the best subset ( $B$ ). Then, for each feature  $f$ , we could calculate the category utility of the concept hierarchy generated using the feature set  $B / \{f\}$ . It seems plausible that the feature whose deletion causes the greatest drop in category utility is the most important. This approach would leave us with an ordering of the features, which is exactly what we need.

However this theory has not been tested, as the evaluation function is computationally intensive; at each step, the concept structure must be reconstructed from scratch using the new feature set  $B / \{f\}$ , and the category utility metric computed. If we used AICC [7] as the concept learner instead of COBWEB, we could add features to the concept structure without having to rebuild the hierarchy from scratch. However this approach would still require too much computation to be suitable for choosing the most predictive feature on the fly.

### 3.2 Principal Components Analysis and Variance

Principal Components Analysis (PCA) is a well-established statistical technique, which has been applied to many different tasks. Since it considers only the features

---

<sup>3</sup> Each *set of siblings* in the hierarchy is referred to as a *partition*. The first partition refers to the children of the root.

and not the classes to which they belong, it can be used for unsupervised feature selection. PCA determines the linear transformation of a sample of points in  $n$ -dimensional space that exhibits the properties of the sample most clearly along the coordinate axes. These new axes are linear combinations of the original axes, found by calculating the eigenvectors of the original data's covariance matrix. The variance along each of these new axes is given by the corresponding eigenvalue in the covariance matrix. Along the new axes the sample variances are extremes, and those which show little or no spread (minimum variance) indicate interdependence in the data. Therefore setting a minimum bound on the amount of variance along each axis is a way of reducing dimensionality [2].

This method thus examines the variance along the new axes to determine those that are unimportant and therefore can be removed. Therefore we decided to determine if variance is an indicator of the relative importance of features. Using PCA, we could ascertain the new axis with the highest variance directly. However we are interested in determining the most important feature in the *original* feature space. Since the new axis with the highest variance is a linear combination of the original feature set, we simply need the feature that dominates this linear combination.

However, simply determining the variance of each of the features in the original data set would have much lower computational cost and would probably be just as valid. Therefore we investigated using a feature's variance in the current set of cases as its evaluation function. Intuitively, it makes sense that a feature with high variance will find fewer possible matches than one with low variance, and therefore would speed up the retrieval process (see Fig. 1). Although this approach lacks a strong theoretical justification, it is worth comparing to the more „rigorous“ methods. We show some results of this comparison in section 4.

### 3.3 Entropy-Based Feature Ranking

The method proposed by Dash and Liu [5,6] seemed the most promising. It uses an entropy measure to rank the features in order of importance - allowing us to access the most important feature directly. In addition, this entropy measure does not need class information to evaluate the features, unlike the information gain measure.

This approach utilises the observation (from entropy theory [10]) that entropy is low for orderly configurations (where there are distinct clusters), but higher for disorderly or „chaotic“ configurations. This is due to the fact that there are few orderly configurations compared to the number of disorderly ones – making the probability of a disorderly configuration so much higher. In other words, it is less likely that instances in the space of observations will be very close or very far (forming distinct clusters), and more likely that instances would be separated by the mean of all distances in the space. Therefore the entropy measure used gives a low score (close to 0.0) to very close or very distant pairs of instances, and a high score (close to 1.0) to those instances separated by a distance close to the mean distance. The total entropy is calculated over *all pairs* of cases as follows:

$$E = - \sum_{i=1}^N \sum_{j=1}^N (S_{ij} \times \log S_{ij} + (1 - S_{ij}) \times \log (1 - S_{ij})) \quad (1)$$

where  $S_{ij}$  stands for the similarity of cases  $i$  and  $j$ .

The algorithm is straightforward. Each feature  $f$  is removed from the feature set FS in turn, and the entropy of the feature set  $FS / \{f\}$  is calculated. The feature whose deletion produces the greatest drop in entropy is the least important, as removing it made the clusters more distinct (indicated by lower entropy). The converse is true for the most important feature. Using this knowledge, the features can be ranked. A further step to the algorithm uses this information to choose the optimal subset of the original features. The method has been shown to be successful in detecting irrelevant features, but cannot detect redundancy in features [6].

This method is directly applicable to the question selection task. Since the initial stage of the algorithm results in an *ordered* list of the features, we can simply implement this part of the algorithm and then query the user with the first feature in the list. Therefore this is the algorithm we chose to implement for our experiments. However, the computational cost of this evaluation function is fairly high. When testing each feature, we need to re-calculate the entropy after removing that feature from the feature set. This involves calculating the sum of the similarities of *every pair* of cases in the case-base, along all dimensions but the removed one. This method is computationally too expensive to generate refining questions at runtime. In our implementation, we improved efficiency by calculating the *total* similarity (along all dimensions) of every pair of cases in the case-base in an initial start-up step. Then each time we wished to calculate the entropy of a particular feature, we simply needed to calculate the similarity of each pair of cases along that particular dimension, and delete this value from the total similarity score for that pair of cases.

This modification did improve the efficiency of the measure, however the information gain metric is still clearly more efficient. Nonetheless, it was important to determine which method was most effective at reducing the number of query questions. Therefore we needed to compare the three metrics experimentally. The results of these experiments are discussed in the following section.

## 4 Experimental Comparisons

### 4.1 Initial Experiments on Data with Known Classes

We first compared the three metrics using data sets that actually contained class information. Our experiments followed the leave-one-out strategy. This strategy removes each case in turn from the case-base and uses it as a target case. Thus the incremental retriever was run  $N$  times for each data set (where  $N$  is the number of cases). This process was repeated for each of the three feature selection techniques; i.e., feature variance, Dash-Liu, Info-Gain + Clustering and a fourth time using random feature selection. The objective was to find the technique that asked the least questions. The results for each data set and each technique were averaged over the  $N$  runs.

In this first set of experiments, refining questions were generated until

- 1. All cases retrieved have the same classification - *or*
- 2. There are no more *discriminating* features left unspecified (i.e. all cases retrieved have the same values for the remaining features) - *or*
- 3. All features have been asked

The graph in Fig. 2 shows the results of experiments on five different data sets. The data sets are all taken from the UCI Machine Learning Repository. In each experiment with a particular data set, the average number of features asked was calculated, and this was then converted to a percentage of the total features in that data set. Since the information gain metric is the only one with access to class information, the comparison might seem unfair. Therefore we took the data sets, ran the clustering algorithm on them, and performed the information gain experiments using these clusters rather than the „correct“ class information.

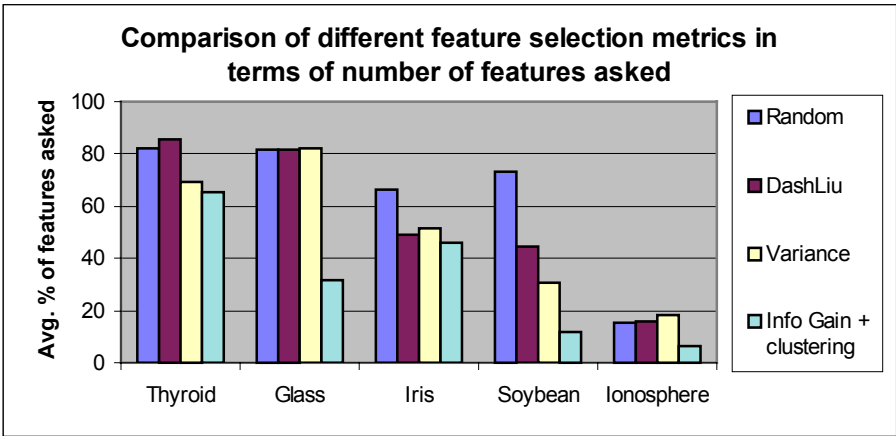


Fig. 2. Comparison of different metrics in terms of number of features asked

As can be seen in Fig. 2, the information gain metric, even in conjunction with clustering, outperforms the other metrics in terms of the number of features needed to perform retrieval, particularly with the soybean data set. It is also interesting that the simple approach of asking features in order of variance characteristics is at least as good, and sometimes better, than the (theoretically more sound) DashLiu metric. Therefore this would seem a good choice of measure for use with unlabelled data as it is the simplest metric and has the lowest computational cost. However we can only calculate the variance of interval-scaled and ordinal (ordered symbolic) data, and many data sets contain categorical/nominal data, therefore it is restricted in its applications. In the Conclusions we discuss the possibility of using entropy as a measure of diversity for category variables that might overcome this problem.

The reason for the poor performance of the DashLiu metric, particularly with the soybean and thyroid data sets, is not clear. Perhaps this can be explained by the fact that this measure has been shown to be unable to detect redundancy [6], but there is

no definite evidence of redundant features in these data sets, so we cannot be sure. However, the particularly bad performance of random selection on the soybean data set would seem to suggest the existence of a lot of irrelevant - or perhaps redundant - features in that data set.

In general, all of the metrics (except the DashLiu measure in one instance, and the variance in another) produce better results than asking the features in random order, which shows they all make a contribution to minimising the inputs needed for retrieval. However it is clear that the information gain metric would be the best choice to assist an online decision guide, as it requires the fewest input features to reach a manageable set of cases.

## 4.2 Experiments with Unlabelled Data

The unsupervised data we used in our experiments was a case-base of laptop computer descriptions comprising over 600 cases. This data was a perfect example of the kind of data that would be used with an online decision guide. In fact, we compare the performance of our system (using the information gain metric) to that of an existing online laptop decision guide, on the personallogic site ([www.personallogic.com](http://www.personallogic.com)). The personallogic system allows customers to impose constraints, which are then exploited by a constraint satisfaction engine in order to prune alternatives that do not satisfy them. Our goal was to prove that our system could retrieve a manageable set of cases with fewer questions. In other words, we wish to show that an adaptive intelligent approach to question order works better than a predetermined one.

Before applying our algorithms for clustering and measuring information gain to the laptop cases they had to be modified to deal with unknown values since there were many missing values in the case descriptions. There are a few different approaches advocated in the literature for dealing with missing values in data (see [13] and [9] for a discussion). Dealing with missing values entails compromises and the strategy adopted in this work is outlined in [8].

Next we had to decide on the best stopping criterion to use when generating our refining questions. With a decision guide the user generally wishes you to „whittle down“ the possibilities into a manageable set, therefore the size of the retrieved set would seem a sensible stopping criterion. Consequently this was used in the experiments instead of those listed in section 4.1. We chose 20 cases as the cut-off figure in our experiments, but in an online situation the user could choose their own preferred maximum.

Similar experiments to those described in section 4.1 were performed on the laptop data. However since the data set contains nominal data we could not use the variance metric, and as it contains unknown values, we could not use the DashLiu metric either, as it does not allow for them. Therefore we simply compared the information gain metric to random generation of refining questions, to ensure the metric was producing useful feature orderings using the detected clusters. Our results showed that, using the information gain metric, on average only 4.66 (or 33%) of the 14 available features were needed for retrieval. Ordering the questions randomly



required on average 7.69 (or 55%) of the features. This result would suggest that the information gain metric works well with the laptop data set.

In order to compare our system to the personallogic system, we chose 20 test cases at random from our case-base as hypothetical targets to guide our input to both systems. For each target case we input answers to the personallogic system in the order asked, stopping when the retrieved set had been cut down to 20 cases or less. The experiments with our system followed the same procedure, but were performed automatically rather than manually. The number of input features necessary to achieve this „manageable set“ differed depending on the target case. The ordering of questions in the personallogic system did suit some of the cases used as target cases. However in all cases bar one, our system performed as well or better.

In some of the experiments, our system stopped asking questions *before* it reached the threshold of 20 retrieved cases. This happened in two different situations. Firstly, if all of the retrieved cases are of the same class, then the *entropy* of the retrieved set (which is calculated as part of the information gain metric) is 0, meaning no more information is needed in order to know the classification of an arbitrary case in the set. Therefore it does not make sense to ask any more questions. The second situation occurs when all of the retrieved cases have the same values for each of the as-yet-unknown features. Therefore asking about these features will be of no use, and is reflected in an *information gain* value of 0 for each of the features. When these situations occur the retrieval algorithm naturally stops early, as there is nothing to be gained from asking further questions (and no way of calculating the order in which to ask them). Therefore it is not always possible for the retriever to continue asking questions until the threshold amount of retrieved cases has been reached. In this situation the cases are returned in the ranked order indicated by the similarity measure described in section 2. In the results shown in Fig. 3, the threshold of 20 was reached in 7 out of the 20 experiments. These 7 are marked, as they are the points of most direct comparison between the two systems. It can be seen that our system performs better in all of those experiments.

## 5 Conclusions and Future Work

The above experiments show that the I-CBR based technique is indeed a powerful mechanism for reducing the number of features needed for retrieval. The fact that this metric requires the imposition of a cluster structure on unlabelled data has proved not to be a problem as it still out-performs the alternatives. The approach would work for a variety of selection problems, as it requires only that the data be represented as features and that it can be clustered. However the key advantage of the incremental approach, whichever metric we use for ordering the questions, is that it *adapts to the user* at run-time. The ordering determined relies entirely on the values entered by the user. In experiments to compare the number of features asked when using this adaptive approach, to those asked using the initial feature ordering calculated over the entire case-base, our dynamic approach not surprisingly performed better. With the laptop case-base, the dynamic approach asked on average 33% of the features, whereas the static approach required 51% of the features to reach an answer.

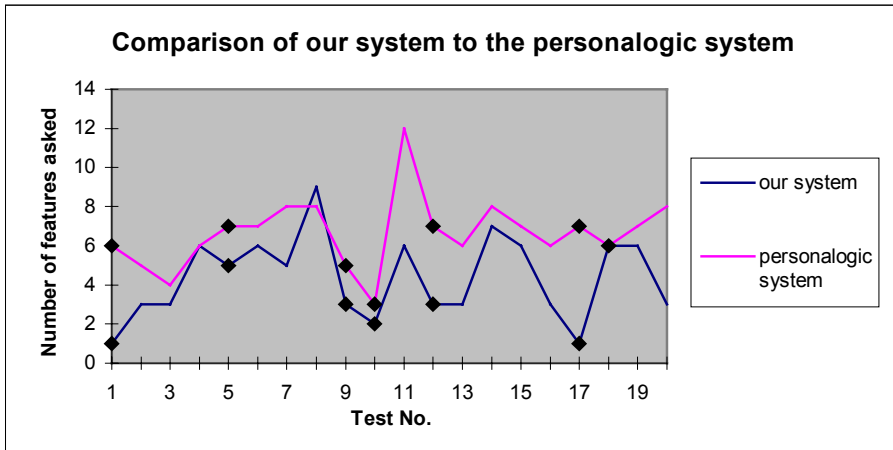


**Fig. 3.** Comparison of our system to the personalogic system in terms of number of features asked – marked points indicate tests in which our system retrieved 20 cases or less and therefore are more directly comparable to the personalogic results

Therefore our system would be an ideal basis for an online decision guide. This could be a product selection site for a single company or a broker-type decision guide like the personalogic site. Since the goal of sites promoting e-commerce is to entice the user to buy, many are using intelligent guides to navigate the user through the products. However having to go through a lengthy interaction in order to find a suitable product may turn many users off the idea. Therefore providing focus to the questioning, which leads to shorter interaction time, would make the whole process a lot more attractive to users.

There is also the added benefit of reduced network request-response cycles. With the explosive growth of the Internet have come problems of increased server load and network latency. This means that systems that require interchange of data between server and client over the network can be slow to unusable (at peak times). The work described here is part of a larger project, which is investigating methods for developing distributed CBR systems that *minimise* the load on the servers and the network, in order to increase response time and usability. Shorter-lived queries mean reduced network traffic, and so this work is complimentary to the project as a whole.

## 5.1 Future Work

One criticism of this approach might be that this method of deciding which question to ask at each stage may result in an „unnatural“ ordering of questions, since human intuition on the relative importance of the features is not taken into account. To address this, our interface allows the user to choose to input a value for whichever feature they wish, if they are not interested in the feature they are prompted for. In

the future we propose to perform some user trials to determine if the question ordering appears reasonable or counter intuitive.

Another issue that warrants exploration is the promising performance of the simple variance metric. So far we have not pursued this approach because of the problem of measuring variance for category variables. However entropy is known to be a good measure of mixture for category variables [4] and we propose to evaluate an extension to the variance metric that incorporates entropy.

## References

1. Aha D., Bankert R. (1995) *A Comparative Evaluation of Sequential Feature Selection Algorithms*, in proceedings of AI & Statistics Workshop 1995.
2. Bishop C. M. (1995) *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.
3. Cunningham P., Smyth B. (1994) *A Comparison of model-based and incremental case-based approaches to electronic fault diagnosis*, in proceedings of the Case-Based Reasoning Workshop, AAAI-1994.
4. Cunningham P., Carney, J. (2000) *Diversity versus Quality in Classification Ensembles based on Feature Selection*, to be presented at European Conference on Machine Learning, Barcelona Spain, June 2000.
5. Dash M., Liu H., Yao J. (1997) *Dimensionality Reduction for Unsupervised Data*, in proceedings of IEEE International Conference on Tools with AI (TAI-97), pp. 532-539.
6. Dash M., Liu H. (1999) *Handling Large Unsupervised Data via Dimensionality Reduction*, in proceedings of SIGMOD Data Mining and Knowledge Discovery Workshop, (DMDK), Philadelphia, USA, May 1999.
7. Devaney M., Ram, A. (1997) *Efficient Feature Selection in Conceptual Clustering*, in proceedings of the 14<sup>th</sup> International Conference on Machine Learning, Nashville, 1997.
8. Doyle M., Cunningham P. (2000) *A Dynamic Approach to Reducing Dialog in On-Line Decision Guides*, (extended version of this paper) Trinity College Dublin Computer Science Technical Report TCD-CS-2000-14, <http://www.cs.tcd.ie/>.
9. Everitt B.S. (1993) *Cluster Analysis*, 3<sup>rd</sup> Ed., Edward Arnold.
10. Fast J. D. (1970) Entropy: the significance of the concept of entropy and its applications in science and technology, chapter 2: *The Statistical Significance of the Entropy Concept*. Philips Technical Library, Eindhoven.
11. Fisher D. H. (1987) *Knowledge Acquisition via incremental conceptual clustering*, in Machine Learning, Vol. 2., pp.139-172.
12. Kaufman L., Rousseeuw P. (1990) *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York.
13. Quinlan J. R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann.
14. Smyth B., Cunningham P. (1994) *A Comparison of Incremental Case-Based Reasoning and Inductive Learning*, in proceedings of the 2<sup>nd</sup> European Workshop on Case-Based Reasoning, Chantilly, France, November 1994.

# Flexible Control of Case-Based Prediction in the Framework of Possibility Theory

Didier Dubois, Eyke Hüllermeier, and Henri Prade

Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier  
`{dubois,eyke,prade}@irit.fr`

**Abstract.** The “similar problem-similar solution” hypothesis underlying case-based reasoning is modelled in the framework of possibility theory and fuzzy sets. Thus, case-based prediction can be realized in the form of fuzzy set-based approximate reasoning. The inference process makes use of fuzzy rules. It is controlled by means of modifier functions acting on such rules and related similarity measures. Our approach also allows for the incorporation of domain-specific (expert) knowledge concerning the typicality (or exceptionality) of the cases at hand. It thus favors a view of case-based reasoning according to which the user interacts closely with the system in order to control the generalization beyond observed data. Our method is compared to instance-based learning and kernel-based density estimation. Loosely speaking, it adopts basic principles of these approaches and supplements them with the capability of combining knowledge and data in a flexible way.

## 1 Introduction

A close connection between fuzzy set-based (approximate reasoning) methods and the inference principle underlying case-based reasoning (CBR) has recently been described [7,17]. Besides, some attempts at combining CBR and methods from fuzzy set theory have already been made (e.g., [13]). This paper continues this promising line of research. It is argued that fuzzy rules in conjunction with associated inference procedures provide a convenient framework for modelling the CBR hypothesis which suggests that “similar problems have similar solutions” [1], and for supporting the inference part of the CBR methodology.

Fuzzy rules provide a local, rough and soft specification of the relation between variables  $X$  and  $Y$  ranging on domains  $D_X$  and  $D_Y$ , respectively. They are generally expressed in the form “if  $X$  is  $A$  then  $Y$  is  $B$ ,” where  $A$  and  $B$  are fuzzy sets associated with symbolic labels and modelled by means of membership functions on  $D_X$  resp.  $D_Y$ .<sup>1</sup> There are several aspects which motivate the use of fuzzy rules in the context of CBR. Firstly, the CBR hypothesis itself can be considered as an *if-then* rule. Secondly, the notion of *similarity*, which lies at the heart of CBR, is also strongly related to the theory of fuzzy sets. Indeed,

---

<sup>1</sup> We use the same notation for a label, the name of an associated fuzzy set, and the membership function of this set.

one of the main interpretations of the membership function of a fuzzy set is that of a similarity relation, i.e., membership values can be thought of as degrees of similarity [8]. Thirdly, linked with the framework of possibility theory, fuzzy sets provide a tool for modelling and processing *uncertainty*. In connection with CBR, this aspect seems to be of special importance if one realizes the *heuristic* character of this problem solving method [11].

The main concern of this paper is to provide a possibilistic framework for modelling the inference process in CBR. More precisely, we make use of possibility rules, a special type of fuzzy rules, in order to formalize the CBR hypothesis.<sup>2</sup> Our focus is on *case-based prediction* [7], which is in line with the idea underlying instance-based learning algorithms [2].

The remaining part of the paper is organized as follows. The basic CBR framework we proceed from is stated in Section 2. In Section 3, we outline the formalization of CBR based on possibility rules. Taking this framework as a point of departure, Section 4 shows how fuzzy set-based modelling techniques can be utilized for translating domain-specific knowledge into a particular CBR model which is suitable for the application at hand. Section 5 reconsiders some aspects of our approach against the background of related methods.

## 2 The Basic Framework

Let a case be defined as a tuple  $\langle s, r \rangle \in \mathcal{C} = \mathcal{S} \times \mathcal{R}$  consisting of a *situation*  $s \in \mathcal{S}$  and an associated *result* or *outcome*  $r \in \mathcal{R}$ .<sup>3</sup> We do not assume that a situation determines a unique outcome, which would be too restrictive for certain applications. That is, cases  $\langle s, r \rangle$  and  $\langle s, r' \rangle$  might be encountered such that  $r \neq r'$ . Let  $\varphi \subset \mathcal{S} \times \mathcal{R}$  denote the class of potential observations. Thus, a case is always an element of the relation  $\varphi$ .

Data is assumed to be given in the form of a memory

$$\mathcal{M} = \{\langle s_1, r_1 \rangle, \langle s_2, r_2 \rangle, \dots, \langle s_n, r_n \rangle\} \quad (1)$$

of precedent cases. Moreover, we suppose the similarity of situations resp. results to be specified by means of (reflexive and symmetric) measures  $\sigma_{\mathcal{S}} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$  and  $\sigma_{\mathcal{R}} : \mathcal{R} \times \mathcal{R} \rightarrow [0, 1]$ .

We consider the task of exploiting past experience (in the form of the memory  $\mathcal{M}$ ) against the background of the CBR hypothesis in order to predict the *result* or *outcome*  $r_0 \in \mathcal{R}$  associated with a new situation  $s_0 \in \mathcal{S}$ . To this end, we shall quantify the possibility that  $r_0 = r$ , i.e., that  $\langle s_0, r \rangle \in \varphi$ , for all outcomes  $r \in \mathcal{R}$ . A prediction of this kind can be used in various ways, e.g., for helping a user in making decisions. It can also support the overall process of case-based reasoning in the sense that it allows subsequent stages of this process to focus

<sup>2</sup> Let us mention that other types of fuzzy rules might be used, and that corresponding models can also be formalized within a more logic-oriented framework [7, 16].

<sup>3</sup> We prefer these expressions for reasons of generality to the commonly used terms “problem” and “solution.”

on the most likely outcomes. For instance, if  $r_0$  corresponds to the solution of a new problem,  $s_0$ , the possibilistic valuation allows one to consider the most promising candidates first, before taking other solutions into account.

For illustration purposes we make use of the AUTOMOBILE DATABASE.<sup>4</sup> This database contains 205 cars, each of which is characterized by 26 attributes. Thus, a case corresponds to a car which is characterized by means of an attribute-value representation including properties such as, e.g., its horsepower and fuel-type. For the sake of simplicity, we shall consider only some of the attributes available, i.e., our memory  $\mathcal{M}$  is actually a projection of the complete database. One of the attributes, namely the price of a car, has been chosen as the outcome associated with a case. The latter is hence a tuple  $\langle s, r \rangle$ , where the situation  $s = (a_1, \dots, a_L)$  is a vector of attribute values describing a car and  $r$  is the associated price.

### 3 Possibility Rules and Case-Based Prediction

A *possibility rule*, subsequently symbolized by  $A \rightsquigarrow B$ , corresponds to the statement that “the more  $X$  is  $A$ , the more *possible*  $B$  is a range for  $Y$ .” More precisely, it can be interpreted as a collection of rules “if  $X = x$ , it is possible at least to the degree  $A(x)$  that  $B$  is a range for  $Y$ .” The intended meaning of this kind of *possibility-qualifying* rule is modelled by the following constraint which guarantees a certain lower bound to the possibility  $\pi(x, y)$  that  $(x, y)$  is an admissible instantiation of  $(X, Y)$ :

$$\pi(x, y) \geq \min\{A(x), B(y)\}. \quad (2)$$

As suggested by the rule-based modelling of the relation between  $X$  and  $Y$ , these variables often play the role of an input and an output, respectively, and one is interested in possible values of  $Y$  while  $X$  is assumed to be given. By letting  $\pi(y|x) = \pi(x, y)$ , the constraint (2) can also be considered as a lower bound to a *conditional* possibility distribution  $\pi(\cdot|x)$ . That is, given the value  $X = x$ , the possibility that  $Y = y$  is lower-bounded by  $\pi(x, y)$ . Note that *nothing* is said about  $Y$  in the case where  $A(x) = 0$  since we then obtain the trivial constraint  $\pi(y|x) \geq 0$ .

A basic idea of the approach discussed in this paper is to use a possibility rule thus defined in order to formalize the CBR hypothesis. In fact, interpreting  $X$  and  $Y$  as degrees of similarity between two situations and two results, respectively, and  $A$  and  $B$  as fuzzy sets of “large similarity degrees” (with strictly increasing membership functions on  $[0, 1]$ ) amounts to expressing the following version of the CBR hypothesis: “The more similar two situations are, the more *possible* it is that the corresponding outcomes are similar” [7]. Note that this formalization takes the heuristic nature of the CBR hypothesis into account. In fact, it does not impose a deterministic constraint, but only concludes on the guaranteed *possibility* that situations yield similar outcomes.

<sup>4</sup> Available at <http://www.ics.uci.edu/~mlearn>.

In the sense of the above principle, an observed case  $\langle s_1, r_1 \rangle \in \mathcal{M}$  is taken as a piece of evidence which qualifies similar (hypothetical) cases  $\langle s, r \rangle$  as being possible. According to (2) it induces lower bounds<sup>5</sup>

$$\pi(s, r) \geq \min \{ \sigma_{\mathcal{S}}(s, s_1), \sigma_{\mathcal{R}}(r, r_1) \} \quad (3)$$

to the possibility that  $\langle s, r \rangle \in \varphi$ . This can be interpreted as a similarity-based *extrapolation* of case-based information: The observation  $\langle s_1, r_1 \rangle$  is considered as a typical case or, say, prototype, which is extrapolated in accordance with the CBR hypothesis. The more similar  $\langle s, r \rangle$  and  $\langle s_1, r_1 \rangle$  in the sense of the (joint) similarity measure  $\sigma_{\mathcal{C}} : (\langle s, r \rangle, \langle s', r' \rangle) \mapsto \min \{ \sigma_{\mathcal{S}}(s, s'), \sigma_{\mathcal{R}}(r, r') \}$  are, the more plausible the (hypothetical) case  $\langle s, r \rangle$  becomes and, hence, the larger is the (lower) possibility bound (3).

Applying (3) to all cases in the memory  $\mathcal{M}$  we obtain the possibility distribution  $\pi_{\mathcal{C}}$  defined by

$$\pi_{\mathcal{C}}(s, r) = \max_{1 \leq i \leq n} \min \{ \sigma_{\mathcal{S}}(s, s_i), \sigma_{\mathcal{R}}(r, r_i) \} \quad (4)$$

for all  $c = (s, r) \in \mathcal{S} \times \mathcal{R}$ . This distribution can be interpreted as a possibilistic approximation of the relation  $\varphi$  (cf. Section 2). It is of provisional nature and actually represents lower bounds to possibility degrees (the equality in (4) is justified by a principle of *maximal informativeness*). In fact, the degree of possibility  $\pi_{\mathcal{C}}(c)$  assigned to a case  $c \in \mathcal{C}$  may increase when gathering further evidence by observing new sample cases, as reflected by the application of the maximum operator in (4). Note that the possibility of a case  $\langle s, r \rangle$  is determined by the maximally similar case in the memory. Thus, similarity-based indexing techniques can be used for computing the possibility degrees (4).<sup>6</sup>

The distribution (4) can be taken as a point of departure for various inference tasks. Particularly, given a new situation  $s_0$ , a prediction of the associated outcome  $r_0$  is obtained in the form of the conditional distribution  $\pi_{\mathcal{R}} = \pi_{\mathcal{C}}(s_0, \cdot)$ :

$$\pi_{\mathcal{R}}(r) = \pi_{\mathcal{R}}(r | s_0) = \pi_{\mathcal{C}}(s_0, r) \quad (5)$$

for all  $r \in \mathcal{R}$ . It is quite interesting to compare this prediction resp. the underlying extrapolation principle to related instance-based methods, especially the *k*-NEAREST NEIGHBOR classification rule (cf. Section 5).

Consider the example introduced in Section 2, and suppose a car to be characterized by only three attributes, namely its horsepower, engine-size, and price. Thus, the CBR hypothesis should simply be understood in the sense that “cars with similar horsepower and similar engine-size have similar prices.” Let the respective elementary similarity relations be given by  $\sigma_{hp} = \sigma_{es} = f_{100}$  and  $\sigma_{pr} = f_{10000}$ , with  $f_M : (x, x') \mapsto \max\{1 - |x - x'|/M, 0\}$ . When modelling the

<sup>5</sup> Without loss of generality, we assume the membership functions of the fuzzy sets of “large similarity degrees” to be given by the identical function  $\text{id} : x \mapsto x$  on  $[0, 1]$ .

<sup>6</sup> In general, only some of these degrees have to be derived.

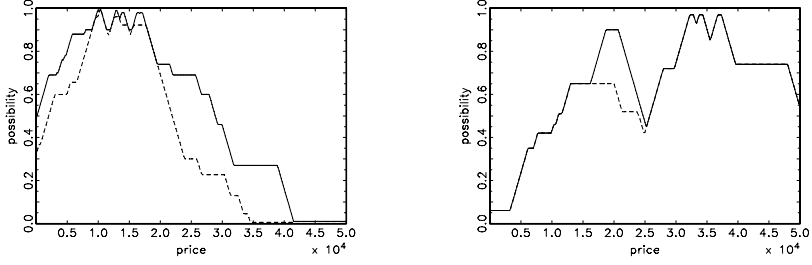


**Fig. 1.** Left: Prediction (5) of the price of a car based on the original hypothesis (solid line) and its modified version (7). Right: Prediction (5) for a car with horsepower 210 (solid line) and the prediction obtained by discounting the 106th case (dashed line)

logical “and” in the above rule by means of the minimum-operator<sup>7</sup> we obtain the measures  $\sigma_S : ((x, y), (x', y')) \mapsto \min\{\sigma_{hp}(x, x'), \sigma_{es}(y, y')\}$  and  $\sigma_R = \sigma_{pr}$ . Figure 1 (left) shows the prediction (5) for  $s_0 = (100, 110)$ . This prediction corresponds to the “more or less” possible range of prices for the class of cars with horsepower of 100 and engine-size of 110. As can be seen, the evidence contained in the database strongly supports prices between \$10,000 and \$17,000.

## 4 Controlling the Inference Process

The possibilistic extrapolation of case-based information in the sense of (3) relies on the heuristic assumption underlying CBR. It should, therefore, take into account whether the related reasoning principle is actually valid. That is, the less the current application seems to meet the CBR hypothesis, the more cautious one should be when considering an observed case as evidence for the existence of similar cases. In this section, we extend the basic model of Section 3 in order to modulate the extent of case-based extrapolation. To this end, we make use of modelling techniques which allow a user to control the way in which the CBR principle is realized.<sup>8</sup>

**Modified Possibility Rules.** The basic idea is to generalize the concept of a possibility rule by making use of so-called *modifiers* [18] in (4), i.e., non-decreasing functions  $[0, 1] \rightarrow [0, 1]$ . Given modifiers  $m_{11}, \dots, m_{1L}$  and  $m_2$ , we consider rules of the form  $m_1 \circ A \stackrel{m_2}{\rightsquigarrow} B$ , where  $m_1 = (m_{11}, \dots, m_{1L})$ . Such a

<sup>7</sup> This operator is a usual choice. Yet, it might be replaced by other conjunction-like combinations.

<sup>8</sup> This knowledge-based approach contrasts with data-driven methods of case-based inference which formalize the CBR principle in a more implicit way (e.g. [15]).



rule gives rise to the following possibility distribution:

$$\pi_C : (s, r) \mapsto \max_{1 \leq i \leq n} m_2 \left( \min \{ m_1(\sigma_S(s, s_i)), \sigma_{\mathcal{R}}(r, r_i) \} \right), \quad (6)$$

where  $m_1(\sigma_S(s, s')) = \min\{m_{11}(\sigma_1(a_1, a'_1)), \dots, m_{1L}(\sigma_L(a_L, a'_L))\}$  for two situations  $s = (a_1, \dots, a_L)$ ,  $s' = (a'_1, \dots, a'_L)$ , and the  $\sigma_j$  ( $1 \leq j \leq L$ ) are elementary similarity measures defined for the individual attributes.

Both modifiers in (6) control the extent to which other (hypothetical) cases are supported by an observation. The “larger” the functions  $m_1$  and  $m_2$  are, the stronger (in the sense of asserted possibility degrees) a sample case  $\langle s_i, r_i \rangle$  is extrapolated, i.e., the stronger hypothetical cases are supported.

More precisely,  $m_1 \circ A \stackrel{m_2}{\rightsquigarrow} B$  stands for the rule that “for  $m_1$ -similar situations it is  $m_2$ -possible that the respective results are similar.” In connection with the linguistic modelling of fuzzy concepts, modifiers such as  $x \mapsto \neg x^2$  or  $x \mapsto \neg \sqrt{x}$  are utilized for depicting the effect of linguistic hedges such as “very” or “almost” [18]. Applying the modifier  $x \mapsto \neg x^2$  in our above example might thus be seen as replacing the original hypothesis by the weaker assumption that “cars with *very* similar horsepower and *very* similar engine-size have similar prices.” The modifier  $m_2$  acts on the possibility-qualifying part of a rule: “ $m_2$ -possible” stands for expressions such as, e.g., “more or less possible” which basically cause a discounting of the rule  $m_1 \circ A \rightsquigarrow B$ . For instance, by taking  $m_2$  as the mapping  $x \mapsto \min\{x, 1 - \lambda\}$ , the distribution (6) becomes

$$\pi_C : (s, r) \mapsto \max_{1 \leq i \leq n} \min \{ 1 - \lambda, \min \{ m_1(\sigma_S(s_0, s_i)), \sigma_{\mathcal{R}}(r, r_i) \} \}. \quad (7)$$

The constant  $\lambda$  plays the role of a discounting factor and defines an upper bound to the support that can be provided by the rule. Particularly, the latter becomes completely ineffective if  $\lambda = 1$ .

Consider the hypothesis that “it is completely possible that cars with *very* similar horsepower and *very* similar engine-size have similar prices” as an example. Applying the modifier  $m_1 : x \mapsto \neg x^2$  to the similarity relations  $\sigma_{hp}$  and  $\sigma_{es}$ , and modelling the (non-)effect of “completely” by  $\lambda = 0$ , the prediction  $\pi_{\mathcal{R}} = \pi_C((100, 110), \cdot)$  based on (7) yields the possibility distribution which is shown by the dashed line in Figure 1 (left). Compared to the prediction (5), the degree of possibility is smaller for most of the prices  $r \in \mathcal{R}$ . This is caused by the fact that the CBR hypothesis is now modelled in a more cautious way. Loosely speaking, cases which are less similar to the current situation are now ignored more strongly.

**Discounting of Cases.** Given a possibility rule expressing the CBR hypothesis, each observed case  $\langle s, r \rangle$  leads to an *instantiation* of this rule. The meta-rule “similar situations possibly have similar outcomes” thus becomes a specific rule related to  $\langle s, r \rangle$ : “If a situation is similar to  $s$ , the corresponding outcome is possibly similar to  $r$ .” This way, the memory (1) of cases implicitly induces a set of  $n$  rules.

Of course, the rule related to a case  $\langle s, r \rangle$  and, hence, the induced extrapolation of  $\langle s, r \rangle$  might be misleading if this case is rather “untypical” or “exceptional.” In our database, for instance, the 106th car has a rather low price (\$19,699), at least in comparison with its horsepower (200). Thus, applying the simple rule that “cars with similar horsepower have similar prices” in connection with this case might provide unjustified support of low prices (for cars with similar horsepower) in further predictions, as exemplified in Figure 1 (right).

Methods for discounting a prediction can be used in order to prevent untypical cases from providing unjustified support of other (hypothetical) cases. Suppose that a “level of typicality” is assigned to each case by some expert. Thus, a case can be thought of as a triple  $\langle s, r, t \rangle$  with  $0 \leq t \leq 1$ . (Numerical levels might be associated with linguistic labels such as, e.g., “typical” or “completely exceptional.”) Let  $\delta_t : [0, 1] \rightarrow [0, 1]$  denote a discounting (i.e., non-increasing) function which depends on the level  $t$  of typicality, i.e.,  $\delta_t \leq \delta_{t'}$  for  $t \leq t'$ . The prediction (6) can then be generalized as follows:

$$\pi_C : (s, r) \mapsto \max_{1 \leq i \leq n} m_2 \left( \delta_{t_i} \left( \min \left\{ m_1(\sigma_S(s_0, s_i)), \sigma_R(r, r_i) \right\} \right) \right). \quad (8)$$

This mapping reveals the combination of two types of discounting, namely that of complete rules (via  $m_2$ ) and that of individual observations (via  $\delta_{t_i}$ ). Figure 1 (right) shows the prediction (8) for the special case where  $\delta_t$  is modelled as  $p \mapsto \min\{p, t\}$ ,  $m_2$  is the identity  $x \mapsto x$ , and (only) the 106th case is discounted by letting  $t_{106} = 1/2$ .

Note that the possibility scales in Figure 1 are ordinal in nature. This means that only the ordering of the levels is important, and that linguistic labels can be attached to numbers (e.g., 1 = “fully possible”, 0.8 = “rather possible”, ..., 0 = “impossible”). Likewise, the level  $t$  of typicality (where  $t = 1$  means “typical,”  $t = 0.8$  means “more or less typical,” ...) belongs to a normality scale which commensurates with the possibility scale.

The adaptation of rules and the discounting of individual cases provide two examples of using fuzzy set-based modelling techniques in order to control the case-based inference process. This framework can be extended further. For instance, the CBR hypothesis can be modelled by means of a *combination* (conjunction) of several rules, where each rule refers to different attributes or modifiers in order to express a condition which allows to conclude on the possibility of outcomes to be similar. One might say, for instance, that cars with very similar horsepower possibly have similar prices, and that similar prices are more or less possible for cars with similar engine-size and approximately similar peak-rpm. We refer to the companion paper [12] for a discussion of further examples and extensions.

**Integration of Background Knowledge.** Interestingly enough, our framework is also well-suited for incorporating background knowledge of more general nature. This becomes especially apparent if such knowledge is also expressed in terms of fuzzy rules. For instance, an expert might be willing to agree that “a

price of slightly more than \$40,000 for a car with horsepower of approximately 200 is completely possible.” This can be formalized as a possibility rule  $A \rightsquigarrow B$ , where  $A$  and  $B$  model the fuzzy sets of “approximately 200” and “slightly more than \$40,000.” Such a rule can simply be added to the rule base induced by the memory of cases, thereby supplementing the “empirical” evidence which comes from observed cases.

A special type of (rule-based) background knowledge is obtained by realizing a kind of hypothetical reasoning. One might specify, for instance, a hypothetical car by means of some attribute values and then ask an expert for a typical (or possible) price. The hypothetical case thus defined can principally be treated in the same way as an observed one. Hypothetical reasoning provides a convenient way of improving sparse memories. It is also interesting from a knowledge acquisition point of view. Indeed, from a user perspective it might appear less difficult to give some specific examples (e.g., by estimating prices of hypothetical cars) than to specify universally valid rules.

Apart from fuzzy rules, more general types of constraints can be used for modelling background knowledge. For instance, in order to satisfy the (reasonable) convexity constraint

$$r' \leq r \leq r'' \Rightarrow \min\{\pi(s, r'), \pi(s, r'')\} \leq \pi(s, r) \quad (9)$$

expressing that intermediary prices of a car should be at least as possible as more extreme prices, a prediction  $\pi_{\mathcal{R}}$  is simply replaced by its convex hull.

## 5 Relations to Other Fields

The methodological framework outlined above will subsequently be referred to as PEC (Possibilistic Extrapolation of Cases). It is related to various methods from other fields such as, e.g., (fuzzy) rule-based reasoning, possibilistic data analysis, instance-based learning (IBL) and non-parametric statistical inference. In this section, we shall briefly discuss its relation to IBL and kernel-based density estimation (KDE).

**Instance-Based Learning.** Instance-based learning (IBL) algorithms are incremental variants of the well-known nearest neighbor (NN) algorithm [4]. IBL methods are mostly applied to concept learning but have also been used for predicting real-valued attributes [14]. Assuming that “similar instances have similar classifications,” they perform according to the nearest neighbor rule, i.e., the class of a yet unclassified instance is estimated by the class of the most similar among the given examples. A generalization of this principle, the  $k$ NN rule, derives a prediction from the  $k > 1$  most similar instances. The simplest IBL algorithm, known as IB1 [2], mainly differs from the nearest neighbor algorithm in that it normalizes the (numeric) attribute values of instances (which are characterized by means of an attribute–value representation), processes instances incrementally, and uses a simple method for tolerating missing attribute values.

Since IBL algorithms exploit the concept of similarity in order to predict the class (= outcome) associated with a new instance (= situation), they are closely related to PEC. Still, the two approaches realize quite different principles of case extrapolation. Particularly, the possibilistic extrapolation in PEC avoids two questionable properties of the extrapolation principle as realized by the  $k$ NN algorithm in its basic form.<sup>9</sup> Firstly, an estimation in IBL does not reflect the *distance* of the nearest neighbors. In fact, the class of an instance can be extrapolated to instances which are hardly similar.<sup>10</sup> In order to avoid this effect, it has been proposed to extend the basic NN rule by a reject option (realized in the form of a distance threshold) according to which a classification is refused if the nearest neighbor is not near enough [5]. In PEC, extrapolation of case-based information is *bounded* in the sense that results of situations are extrapolated only to similar situations. More precisely, the support of an outcome is graded according to the respective similarity.<sup>11</sup> For instance, the most plausible outcome,  $r$ , of a new situation  $s_0$  might still be supported by a rather small possibility degree  $\pi_{\mathcal{R}}(r) \ll 1$  if the corresponding nearest neighbor of  $s_0$  is actually quite dissimilar.

Secondly, IBL realizes a point-estimation by predicting a single class label.<sup>12</sup> An estimation of this kind cannot represent the *ambiguity* caused by the existence of many different class labels among the nearest neighbors (which entails a large classification error). Again, the basic NN algorithm can be extended by a reject option in order to account for this problem [10]. By providing a prediction in the form of a possibility distribution, i.e., a more general concept which can support different outcomes to different degrees, ambiguity is reflected in a rather natural way in PEC. In fact, a prediction is ambiguous if there are several plausible outcomes having quite similar degrees of possibility.

Besides, there are some minor differences between IBL and PEC. Here, let us only mention one further point, namely that no similarity measure is defined over the set of classes in IBL. In fact, an instance only supports the class to which it belongs. As opposed to this, a case also supports *similar* outcomes in PEC.

**Kernel-Based Density Estimation.** Consider a set  $\{x_1, \dots, x_n\} \subset \mathbb{R}^m$  of realizations of a random variable  $X$ . A kernel-based estimation of an underlying density function (KDE) is then defined as

$$\phi : x \mapsto \frac{1}{n} \sum_{i=1}^n \kappa_h(x - x_i) = \frac{1}{n} \sum_{i=1}^n \kappa\left(\frac{x - x_i}{h}\right), \quad (10)$$

<sup>9</sup> See [6] for an approach which makes use of belief functions in order to solve these problems.

<sup>10</sup> This generally happens if only few observations have been made.

<sup>11</sup> Extrapolation is also graded in the *weighted* version of the  $k$ NN algorithm [9]. There, however, it depends on its *relative* rather than its *absolute* distance.

<sup>12</sup> Still, the use of the NN principle for (non-parametric) density estimation allows one to consider this estimation in a probabilistic context [3].

where  $\kappa$  denotes the *kernel function*. The so-called *kernel width* or *smoothing parameter*,  $h$ , has an important effect on the accuracy of the approximation (10). Large values of  $h$  lead to the problem of over-smoothing. As opposed to this, (10) tends to represent the structure of the data rather than that of the true density if  $h$  is chosen too small.

The allocation of probability mass by means of kernel functions reflects the same line of thought as the CBR principle. Indeed, the striking similarity between (10) and (5) is revealed by writing the latter as

$$\pi_{\mathcal{C}} : c \mapsto \max_{1 \leq i \leq n} \sigma_{\mathcal{C}}(c, c_i), \quad (11)$$

where  $c_i = \langle s_i, r_i \rangle$  is the  $i$ th case in the memory. In fact, (11) can be seen as the possibilistic counterpart to (10): Instead of taking the average over a number of probability densities, the “possibilistic kernels”  $\sigma_{\mathcal{C}}(\cdot, c_i)$  associated with the data are combined by means of the maximum operator.

Despite the formal resemblance one should, however, not disregard a substantial difference in the semantics underlying (10) and (11). KDE derives estimations of degrees of probability (density) which entail a certain expectation concerning the *frequency* of observations. As opposed to this, PEC is not based on the idea of frequency. Rather,  $\pi_{\mathcal{C}}(c)$  defines a degree of *plausibility*, and this degree increases when similar cases are encountered. Thus, a possibility degree  $\pi_{\mathcal{C}}(c)$  is a quantification of the plausibility that  $c \in \varphi$ , whereas  $\phi(x)$  concerns, not the question of *existence*, but the chance of *occurrence*. Comparing KDE and PEC from a semantical point of view hence reveals that the notions of probability and frequency find their counterparts in terms of possibility and similarity.

Without questioning the usefulness of a density estimation let us mention some points in favor of the possibilistic approach. Firstly, KDE relies on rather restrictive assumptions concerning the generation of data. Consequently, it might produce misleading results if these assumptions are not satisfied, e.g., if observations are not stochastically independent. PEC is obviously more robust in this respect. Secondly, the results of KDE have to be considered with caution if not enough data is available, a problem encountered quite regularly in practice. In fact, non-parametric density estimation is extremely affected by the “curse of dimensionality.”<sup>13</sup> Clearly, PEC is not able to extract more information from the data. The important point, however, is to realize that it does not pretend to do so. On the contrary, PEC does not lay claim to the derivation of a somehow representative distribution. It rather reflects a weaker form of information, since possibility degrees (11) are merely interpreted as *lower bounds*. They can hence represent complete ignorance (as reflected by  $\pi_{\mathcal{C}} \equiv 0$ ) and are in this sense less hypothetical than estimations (10). Thirdly, PEC seems to be more flexible than KDE concerning the integration of knowledge and data, and the combination with other forms of knowledge representation. This is exemplified by the various possibilities of modifying possibilistic kernels by controlling the extrapolation of cases, as discussed in Section 4. The same is true for the integration of general

<sup>13</sup> Not to mention the problem of handling attributes on different scales.

background knowledge. The convexity constraint (9), for instance, can easily be met in PEC but leads to severe difficulties in kernel-based density estimation.

## 6 Summary and Discussion

We have outlined a possibilistic approach to case-based prediction. The basic principle of our method, referred to as PEC, is a similarity-guided extrapolation of observed cases. According to this principle, which relies on the CBR hypothesis and which has been formalized within the framework of fuzzy rules, an already encountered case is taken as evidence for the existence of similar cases. This evidence is expressed in terms of degrees of possibility assigned to hypothetical cases.

PEC has basic features in common with instance-based learning and non-parametric statistical methods, especially kernel-based density estimation. In comparison with these approaches, a main advantage of PEC is the possibility of combining knowledge and data in a flexible way. Firstly, such knowledge plays an important role in modelling the CBR hypothesis, i.e., in controlling the local generalization of sample cases. Secondly, general background knowledge can supplement case-based information when it comes to making predictions. A prediction in the form of a possibility distribution thus results from the combination of several ingredients, namely the observed cases, the (heuristic) “CBR knowledge” which dictates how to generalize beyond this data, and background knowledge which supplements or modifies the extrapolation.

One of the basic ideas of our approach is that of exploiting the merits of linguistic modelling techniques in the context of CBR. It does not mean, however, that a human expert is expected to come up with an optimal model from the start. Rather, it might be sufficient if he specifies a broad structure in a first step, including, e.g., the selection and combination of important attributes which appear together in a rule. A corresponding rule base can then be calibrated afterwards. Firstly, one might apply methods from machine learning in order to assign a mathematical function to each linguistic modifier.<sup>14</sup> Secondly, assuming all concepts to be predefined, the user itself can adapt the rule base, e.g., by exchanging modifiers or by adding new attributes to a rule. Such modifications will generally be based on the consequences of the current CBR model, i.e., on feedback obtained from corresponding predictions. The critical inspection of the prediction in Figure 1 (right), for instance, might cause the user to classify an observation as an exceptional case, thereby influencing future predictions. From this point of view, our approach to case-based inference should clearly be seen as an interactive process in which the user plays an important role.

PEC can be applied in different ways. For instance, a prediction in the form of a possibility distribution can provide useful information in the context of decision making [7]. In connection with case-based problem solving, it can be used for bringing a set of potential solutions into focus: The estimation of the possibility

---

<sup>14</sup> Of course, this approach also requires a precise optimization criterion.

$\pi_{\mathcal{R}}(r)$  that  $r$  is the solution (= result) of the new problem (= situation)  $s_0$ , or that  $r$  can be adapted in a suitable way, allows one to concentrate on the most promising candidates and, hence, to improve the efficiency of the overall problem solving process.

In summary, the method proposed in this paper contributes some basic concepts to the idea of (possibilistic) case-based prediction and decision support which allows for the integration of knowledge and the interaction with the user in a flexible way. The next step is to realize these concepts in the form of a concrete application.

## Acknowledgements

The authors are grateful to an anonymous referee for detailed and very useful comments which helped to improve the paper.

## References

1. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994. [61](#)
2. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991. [62](#), [68](#)
3. C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995. [69](#)
4. B. V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991. [68](#)
5. B. V. Dasarthy. Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(1):67–71, 1980. [69](#)
6. T. Denoeux. A k-nearest neighbor classification rule based on Dempster-Shafer Theory. *IEEE Trans. Systems, Man, and Cybernetics*, 25(5):804–813, 1995. [69](#)
7. D. Dubois, F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, and H. Prade. Fuzzy set modelling in case-based reasoning. *Int. J. Intell. Syst.*, 13:345–373, 1998. [61](#), [62](#), [63](#), [71](#)
8. D. Dubois and H. Prade. The three semantics of fuzzy sets. *Fuzzy Sets and Systems*, 90(2):141–150, 1997. [62](#)
9. S. A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327, 1976. [69](#)
10. M. E. Hellman. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems, Man, and Cybernetics*, SSC-6:179–185, 1970. [69](#)
11. E. Hüllermeier. Toward a probabilistic formalization of case-based inference. *Proc. IJCAI-99*, pages 248–253, 1999. [62](#)
12. E. Hüllermeier, D. Dubois, and H. Prade. Knowledge-based extrapolation of cases: A possibilistic approach. In *Proceedings IPMU-2000*, 2000. To appear. [67](#)
13. M. Jaczynski and B. Trousse. Fuzzy logic for the retrieval step of a case-based reasoner. In *Proc. EWCBBR-94*, pages 313–321, 1994. [61](#)
14. D. Kibler and D. W. Aha. Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5:51–57, 1989. [68](#)

15. P. Myllymäki and H. Tirri. Bayesian case-based reasoning with neural networks. *Proc. IEEE Int. Conf. Neural Networks*, pages 422–427, 1993. 65
16. E. Plaza, F. Esteva, P. Garcia, L. Godo, and R. Lopez de Mantaras. A logical approach to case-based reasoning using fuzzy similarity relations. *Journal of Information Sciences*, 106:105–122, 1998. 62
17. R. R. Yager. Case-based reasoning, fuzzy systems modelling and solution composition. *Proc. ICCBP-97*, pages 633–643, 1997. 61
18. L. A. Zadeh. A fuzzy-set theoretic interpretation of linguistic hedges. *J. Cybernetics*, 2(3):4–32, 1972. 65, 66



# Partial Orders and Indifference Relations: Being Purposefully Vague in Case-Based Retrieval\*

Alex Ferguson and Derek Bridge

Department of Computer Science, University College, Cork  
{a.ferguson,d.bridge}@cs.ucc.ie

**Abstract.** In this paper, we look at case retrieval systems for product selection. Such systems are interactive. This places demands on the technology: customers must be able to specify their requirements in ways that are meaningful to them; and, the cases that are retrieved must be comprehensible in terms of the customer requirements. To meet these demands, we introduce to case retrieval the notions of similarity metrics with *partially-ordered return types* and of relations that express *indifference between degrees of similarity*.

## 1 Product Selection

Product selection applications are increasingly being built using case retrieval technology [1]. Products are regarded as the cases. Customers specify preferred values for case attributes, and these values are assembled into a *probe* case. Cases in the case base (products) are compared, using a similarity metric, to the probe and the most similar are retrieved and presented to the customer.

Such systems are *interactive*, and this places the following demands:

**The Requirements Criterion:** Customers must be able to specify their requirements in ways that are meaningful to them; and

**The Results Criterion:** Customers must be able to understand, in terms of their requirements, why cases have been included in, or excluded from, the result set.

In this paper, we argue that current case retrieval technology in general does not satisfy either of these criteria particularly well. The problem stems in part from a kind of ‘spurious precision’ arising from the way that numerically-valued similarity metrics are used in these systems.

Our alternative approach recognises that vagueness and imprecision in customer requirements is a fact of life. They arise from a number of sources, including: the inconvenience to the customer of formulating complete and precise requirements; the customers’ lack of knowledge of the product space; and their genuine indifference between different portions of the product space. Whatever the source, we feel it is important to model and use this vagueness explicitly.

---

\* This research was funded by Enterprise Ireland (grant number ST/98/024), with the support of Interactive Multimedia Systems, Dublin.

Our solution is two-fold. First, by use of similarity metrics with partially-ordered return types, we avoid unnecessarily equating quite different notions of similarity. Second, by use of relations that specify indifference between degrees of similarity, we can avoid needlessly distinguishing between different degrees of similarity, when the differences are an artefact of the precision of the similarity metric but are not supported by the customer’s own notions of what makes cases similar enough to be meaningfully included together in the same result set.

## 2 Similarity Metrics

In our framework, similarity metrics,  $\sim$ , are functions of the following type:

$$\sim :: \alpha \rightarrow \alpha \rightarrow P$$

where  $P$  is a partially-ordered set, i.e.  $P = (S, \sqsubseteq)$ . In words, a similarity metric takes one value of type  $\alpha$  and another value of type  $\alpha$  and computes their degree of similarity, which will be represented by a value from some set  $S$  that has a partial order  $\sqsubseteq$  defined on it. The ordering on  $S$  is needed so that we can, e.g., determine when one degree of similarity is higher than another.<sup>1</sup>

In most current case retrieval systems, similarity metrics are actually of type:

$$\sim :: \alpha \rightarrow \alpha \rightarrow ([0, 1], <)$$

i.e. degrees of similarity are denoted by numbers between 0 and 1 inclusive. This is subsumed by our framework.

It is useful to use the more general form when doing research into similarity. Then, outcomes of the research apply not only to conventional numeric-valued similarity metrics but also to other ways of measuring similarity. For example, Plaza proposes that first-order feature structures can denote degrees of similarity [5]. Plaza’s feature-structure-valued functions are subsumed by our framework and so our research applies to his similarity metrics as well as to more conventional similarity metrics.

Our past papers more fully explore the advantages of working in this more general framework, e.g. [4], [2], [3]. In this paper, we will mostly use the more familiar numeric-valued similarity metrics where possible. But we will exploit the power of the more general framework when we come to combine component (attribute-specific) similarity metrics into composite (overall) similarity metrics for whole cases (section 7).

## 3 Choosing the Result Set: ‘best- $n$ ’

Consider a case base of products, one of whose attributes is the price of the product. Assume the knowledge engineer believes that prices will not exceed

---

<sup>1</sup> Note that, in our framework, it is the *result type* that is a partially-ordered set, and not the *cases*.

£1000,  $MAX = 1000$ . If we take  $p$  to be the customer's probe value and  $c$  to be the value of a case's price attribute, then one possible similarity metric is the 'less is better' similarity metric from [1], e.g.:

$$c \sim p \hat{=} \begin{cases} 1, & \text{if } c \leq p; \\ 0.8 \times \frac{MAX-c}{MAX-p}, & \text{if } c > p. \end{cases}$$

Suppose the customer decides to retrieve cases based only on their price similarity. (Subsequent sections of the paper will deal with the situation where customers specify preferred values for more than one case attribute.)

The 'less is better' similarity metric, like most numerically-valued metrics, has a high degree of numeric precision. Quite small differences in the computed degrees of similarity may arise. For example, if the customer's probe price is £600, then cases whose prices are £650 and £651 are similar to the probe to degrees of 0.7 and 0.698 respectively. If one of these two cases is in the result set, then the other probably ought to be as well: it is unlikely that such a small numerical distinction would be regarded as decisive by the customer; to exclude only one of them from the result set would probably not reflect the customer's intent. This is well-recognised in the CBR community. But the methods used to overcome this 'spurious precision' merely finesse the problem.

Perhaps most commonly, case retrieval systems mitigate the problem by presenting to the customer a number,  $n$ , of the highest-scoring cases. This is convenient but not entirely satisfactory.

Suppose  $n$  is 10: the top 10 cases (according to the price similarity metric) will be displayed to the customer. The problem with this approach is that the cases that the customer sees are determined arbitrarily by the choice of  $n$ . For example, the top-ranked case and the second-ranked case might have similarities to the probe that are quite close, e.g. 0.7 and 0.698. They are not decisively different cases, so it feels right that both are in the result set. But the cases ranked 10th and 11th might have equally close degrees of similarity, say 0.2 and 0.198. These are not decisively different cases, and yet, for  $n = 10$ , the 11th-ranked case would not be included in the result set.

It is also possible that there are large discontinuities in the degrees of similarity to the probe of the cases in the result set. For example, the 1st, 2nd and 3rd-ranked cases might all have very close degrees of similarity, e.g. 0.7, 0.698 and 0.696. But suppose the similarity to the probe of the 4th-ranked case is 0.3. This is quite different from that of the 3rd-ranked case. The requirement to include  $n = 10$  cases in the result set dictates that cases such as these, which, from the customer's point of view, are decisively less similar to the probe than the top 3 cases, should nevertheless appear in the result set.

We feel that 'best- $n$ ' is an approach that fails to meet the Requirements and Results Criteria. As part of specifying their requirements, customers or knowledge engineers will have to guess a value for  $n$ . What they want is an  $n$  that will retrieve cases that share a high degree of similarity to the probe. But there is no informed way of choosing this value. Instead, in CBR, we typically use a value that gives a 'screenful' of cases. But this gives the problems described in

the previous paragraph. In interpreting result sets, customers must therefore keep in mind that, because the value for  $n$  is arbitrary, the result set may include cases that are decisively different or exclude cases that are not decisively different.

(Of course, this is not a decisive condemnation of ‘best- $n$ ’. Customers may be comfortable with the idea that the system will always display a screenful of cases; and, the problems with the arbitrariness of what is included and excluded are lessened by making provision for iterative query revision.)

An alternative solution to the problem of ‘spurious precision’ is to use integer division on the prices before computing their similarities.<sup>2</sup> Integer division ‘pre-processes’ the case data so that cases with quite similar attribute values are placed into equivalence classes. For example, suppose we use integer division by 10 on the example prices from earlier. After integer division by 10, the two prices £650 and £651 are both ‘reduced’ to 65, so their degrees of similarity to the probe (originally £600 but also ‘reduced’ to 60) are both 0.7.

At first glance, this appears to be doing the right kind of thing. Cases that are not decisively different are receiving the same degrees of similarity to the probe. But this is illusory.

Consider the following four prices £649, £650, £651 and £659. Using the integer division approach, the similarities are 0.72, 0.7, 0.7 and 0.7 respectively. Some cases with quite similar prices (e.g. the cases whose prices are £649 and £650) receive different degrees of similarity (0.72 and 0.7) at the same time that cases that differ more in their prices (the cases whose prices are £650 and £659) receive the same degrees of similarity (0.7).

In the next section, we present indifference relations as an alternative to ‘best- $n$ ’ and to the use of integer division. When we define ‘indifference’ below, we choose not to require transitivity, precisely so that we avoid the problems mentioned in the previous paragraph.

## 4 Choosing the Result Set: Indifference Relations

We propose the idea of *indifference relations* as an alternative approach that better achieves the Requirements and Results Criteria. Using indifference relations, we can ensure that the cases in the result set are ones that the customer perceives as related. The  $(i + 1)$ th-ranked case will only appear in the result set if its similarity to the probe is, according to the customer, indistinguishable from the similarity to the probe of the  $i$ th-ranked case. On occasion, the result set may be smaller than  $n = 10$ ; on other occasions it may be larger. But, in all cases, inclusion or exclusion will be meaningful and not arbitrary.

Indifference then will be a relation between degrees of similarity. In the price example, it does not compare prices; it compares their similarities. We will denote an indifference relation by  $\simeq$ . Given two degrees of similarity,  $s_1$  and  $s_2$ ,  $s_1 \simeq s_2$  returns **True** iff  $s_1$  and  $s_2$  are not decisively different degrees of similarity. The customer or knowledge engineer will need to define this relation for each

<sup>2</sup> The criticisms we make of integer division on prices apply equally well to truncating or ‘rounding-off’ similarity values.

attribute. Their task is to define under what circumstances a customer would be indifferent between two degrees of similarity for an attribute.

The result set that will be displayed to a customer will no longer be the ‘best- $n$ ’. Suppose the cases that are *most* similar to the probe are similar to the probe to degree  $m$ . The result set will include all cases whose similarity to the probe is  $m$  or ‘close enough’ to  $m$ . The indifference relation tells you whether a degree of similarity is ‘close enough’ to  $m$ .

The formal definition is written in a way that can handle similarity metrics having any partial-order as their return type:

$$\text{Retrieve}(CB, p, \sim, \simeq) \hat{=} \{c : c \in CB, \exists m \in M : c \sim p \simeq m\}$$

$$\text{where } M \hat{=} \max\{c \sim p : c \in CB\}$$

In words, for case base  $CB$ , probe  $p$ , similarity metric  $\sim$  and indifference relation  $\simeq$ , if  $M$  is the set of maximal degrees of similarity to  $p$ , the cases to retrieve are those whose degrees of similarity are sufficiently close to a value in  $M$  (as determined by  $\simeq$ ).

A good example of an indifference relation is  $\epsilon$ -equality, i.e. we are indifferent between two degrees of similarity,  $s_1$  and  $s_2$ , iff  $\text{abs}(s_1 - s_2) < \epsilon$ . (As we explain in section 5,  $\epsilon$ -equality is not an equivalence relation. It thereby avoids the problems we encountered with integer division in section 3.)

Suppose there are cases whose prices are £649, £650, £651 and £659. Their similarities to a probe of £600 are 0.702, 0.7, 0.698 and 0.682 respectively. The case whose price is £649 is the most similar to the probe, so this case is in the result set. If we take  $\epsilon$  to be 0.003, the result set will also include the case whose price is £650. This is in the result set because we are indifferent between the degrees of similarity 0.702 and 0.7 (they are less than 0.003 different). The other cases are not in the result set: their degrees of similarity are not close enough to that of the best case. Of course, if we take  $\epsilon$  to be 0.005, then the case whose price is £651 will also be in the result set.

We believe that using indifference in this way better satisfies the Requirements and Results Criteria. Specifying an indifference relation is meaningful, and it has a meaningful effect on the result set: non-maximally similar cases are included in the result set only if their similarity falls short of the similarity of a maximally similar case by an amount about which the customer is indifferent.

In fact, this whole scenario can be made even more meaningful by using similarity metrics with more meaningful return types, for example:

$$c \sim p \hat{=} -\text{abs}(c - p)$$

— the negation of the absolute difference. When the case value equals the probe value, the degree of similarity is at its highest, i.e. zero. The more dissimilar the case value is from the probe, the lower the similarity, as denoted by negative numbers of larger magnitude. The similarity of £650 to £600 is  $-50$ , which is higher than the similarity of £651 to £600, which is  $-51$ .

In this metric, the return type is not normalised to  $[0, 1]$ . This exploits a little of the power of our similarity metric framework, which allows any partially-ordered set as the return type. But, furthermore, the degrees of similarity are themselves now more meaningful to customers. They are sums of money (negated so that the distance function becomes a similarity function).

Specifying an indifference relation now becomes more meaningful too. A customer who takes  $\epsilon$  to be 5 is saying that s/he wants to see the best cases, but s/he would not wish to exclude a case from the result set for the sake of a difference in price of £5. With only a little domain knowledge, meaningful *context-sensitive* choices for  $\epsilon$  can be made: when looking to purchase a house,  $\epsilon$  might be £2000; when looking to purchase cars,  $\epsilon$  might be £400 for one customer or it might be £800 for another customer; when looking at monthly rents,  $\epsilon$  might be £20 on one occasion and £30 on another occasion. The decision can be made meaningfully by the customer in a customer- and goal-specific way.

The next section makes more precise what we mean by indifference by considering its axiomatisation.

## 5 Axiomatisation of Indifference

In general, if we are to avoid the kinds of problems we encountered when using integer division in section 3, we do not want indifference relations to construct equivalence classes of similarity values. Indifference relations will generally be weaker than equivalence relations. It is reasonable to require that they be reflexive and symmetric, but they will not in general be transitive.<sup>3</sup> Instead, we define a weaker property that plays a similar role to transitivity, and which we refer to as *convexity*. We define a relation  $R$  to be  $P$ -convex over any partial order  $P$  iff:

$$\forall x, y, z \in P : x \sqsubseteq_P y \sqsubseteq_P z, x R z \Rightarrow x R y, y R z$$

Given three ordered ‘points’, if the top and bottom points are related by  $R$ , then the middle point must also be related to both. We require this property to ensure that there are no ‘discontinuities’ in the behaviour of indifference relations.

$\epsilon$ -equality is reflexive and symmetric. It is not transitive: with  $\epsilon = 2$ , we are indifferent between values 3 and 4, and between 4 and 5, but not between 3 and 5. It is not, therefore, an equivalence relation. But, it does satisfy convexity: if  $x \leq y \leq z$  and if  $x$  and  $z$  are within  $\epsilon$  of each other, then  $x$  and  $y$  are within  $\epsilon$  of each other, and  $y$  and  $z$  are within  $\epsilon$  of each other.

## 6 Combining Similarity Metrics: Weighted Averages

For the rest of this paper, we consider the situation where the customer supplies preferred values for several attributes, although, for ease of exposition, we

---

<sup>3</sup> The question of whether *indifference* is, e.g., symmetric is orthogonal to the question of whether *similarity* is symmetric.

consider the case of two attributes. We need to combine the two *individual* judgements into an *overall* judgement. The two *component* similarity metrics need to be combined into a single *composite* metric.

The conventional approach is to take a (weighted) average of the two (numeric) degrees of similarity. Using an unweighted average is supposed to allow the two component metrics to contribute equally to the composite similarity judgement; using weighted metrics with unequal weights is supposed to allow relative importances of the attributes to be taken into account.

Taking an arithmetic average is an attractively simple solution. But it immediately raises the question of whether this produces a composite metric which reflects the intuitively desired properties (such as allowing component metrics to contribute equally or allowing one to be more important than the other), or indeed whether the composite is meaningful in any strong sense at all. In particular, one might be seen to be ‘comparing apples and oranges’; the more differently each metric measures the similarity of its respective attribute, the less clear it is to see whether a particular degree of similarity computed by one metric really equates to a degree of similarity computed by the other metric.

Suppose, for example, that the case base contains hotel descriptions. Our probe price is £60 and our probe comfort-level is ‘3-stars’. Suppose there is a case in the case base describing a ‘1-star’ hotel offering rooms for £58. Using negated absolute difference, its similarity to the probe on price is  $-2$ ; its similarity to the probe on comfort is also  $-2$ . But few customers would regard these two degrees of similarity as directly comparable, and even fewer would regard the matches on the two attributes to have been successful to the same degree.

Of course, such problems are well-recognised. The usual solution is to perform some sort of normalisation; each component metric is post-processed, so as to conform to some similar pattern. But we are going to claim that this finesses the problem and the real solution is to find out from the customer how different degrees of similarity for different attributes can be ‘traded’ against each other.

The simplest approach to normalisation is to range-normalise, typically to the interval  $[0, 1]$ . The return type of the metric is re-scaled so that its minimum possible value is mapped to 0, and its maximum to 1, and the remainder is a linear interpolation between these points. (This is built into the ‘less is better’ metric given in section 3, for example.) One difficulty is immediate: if no clear bounds exist on the return type of the original metric (due to a lack of such bounds on the case data), then some guesswork is required. Price attributes are a common example of this: knowledge engineers have to pick an arbitrary maximum price (and perhaps even revise it in the light of price inflation!).

Somewhat more subtly, the metric that results from range-normalisation depends very much on the range chosen. Specifically, the magnitudes of the degrees of similarity will depend on the range. A knowledge engineer might feel obliged to use a wide range if ‘rogue’ values (extremely high or low values) appear in one or other of the case attributes; other knowledge engineers might remove rogue values from consideration when normalising [7]. In the absence of clear bounds, a cautious knowledge engineer who over-estimates a maximum value will ob-

tain a different metric from a less cautious knowledge engineer. In one metric, a narrow normalisation range may have given coarse degrees of similarity; in the other metric, a wider normalisation range may have given fine-grained degrees of similarity; in such circumstances, it is not reasonable to assume that the ways these metrics measure similarity have been made 'equivalent'. This means that there can be just as much of a problem in comparing degrees of similarity for different metrics, even after range-normalisation.

An alternative to range-normalisation, which may take these difficulties into account, is to re-scale according to statistical properties of a metric, if these are known or can at least be approximated. This approach will typically convert a function with a given mean and standard deviation to a function with a predetermined mean and standard deviation. This is more 'stable' than range-normalisation, but to deploy it also requires somewhat better-quality information than does range-normalisation. But, for product selection applications in particular, there remains a fundamental problem, as discussed in the next paragraph.

An assumption underlying these different methods of normalisation is that 'equivalence' of degrees of similarity from different similarity metrics is at bottom a matter of pro-rating one consideration against another. This is seen as a technical problem: how to determine what this pro-rating factor might actually be (e.g. based on range, based on range but ignoring rogue values, based on statistical measures, etc.). But there is a more fundamental problem: in product selection applications in particular, finding this pro-rating is not just a mathematical exercise; it is also a cognitive exercise. The assumption that a technical, mathematical 'fix' will tell us how a customer would be prepared to 'trade-off' differences in similarity values in different metrics is at least questionable. In product selection applications, where we are concerned with subjective user preferences, the assumption is likely to be wrong.

Suppose we could elicit some notion of metric sensitivity: how strongly the customer feels about a change in one metric's similarity value, as compared with a change in another metric's similarity value. If we could determine this, we could normalise by the same factor. Then, normalisation would not be treating the metrics simply as mathematical objects, but would be taking into account customer perceptions. But, such an approach could involve considerable up-front elicitation groundwork. (And, this burden is made all the greater because, in general, the normalisation factor would be customer- and goal-specific.) For these reasons, in CBR, we tend to take the mathematical approach to normalisation, rather than the cognitive approach described in this paragraph.

Finding the right normalisation is only part of the problem. Averaging brings problems too. A first problem is that combining two numbers into a single number loses information that the customer might have found useful. If the customer could see the judgements of the component metrics, s/he could make more informed product selection decisions or query revision decisions.

A second, related problem is that averaging cannot be *agnostic* in the way it combines even conflicting judgements: the degrees of similarity it obtains are always totally-ordered. If case 1 is more similar to the probe on one attribute



than is case 2 but, at the same time, it is less similar to the probe on the other attribute, then *in the absence of any other information from the customer* (and in the light of the problems of ‘equating’ degrees of similarity in normalisation), agnosticism about the ranking of these two cases would be more appropriate than forcing an ordering onto them.

Finally, if we accept that component metrics exhibit ‘spurious precision’, then a composite metric, formed using averaging, will also exhibit ‘spurious precision’: the precision in the component metrics may not reflect customer perceptions, so the precision of the composite metric is also unjustified (especially if it is even more precise than that of the components).

The weights used in averaging are another problem. Contrary to popular belief, weights are hard to interpret. Setting the weight of one similarity metric to 1 and another to 2 does not, in general, give the second attribute twice the weight of the first. One reason for this is the failure to start from ‘a level playing field’. If normalisation has not adequately equated similarity values from the different similarity metrics, then one or other of the metrics may count for less at the outset, and weighting may simply increase or decrease this initial imbalance.

On top of this, weights have no real meaning in terms of the original similarity metrics. Customers who are asked to choose weights (or indicate attribute importances in some way) can only do so in an uninformed way. They have no clear idea of the effect their weights will have. For example, if a customer is viewing a result set and decides to re-issue the original query but with new weights, the customer has no way of knowing in advance whether the new result set will be identical to the original, a re-ranked version of the original, completely different from the original, or partly the same as the original.

In conclusion, conventional approaches to combining similarity metrics compound at least four levels of arbitrariness and ‘spurious precision’. First, the original metrics are often spuriously precise (as argued earlier in this paper). Second, the normalisation is often arbitrary and aims to satisfy mathematical objectives instead of cognitive ones. Third, the use of averaging compounds the precision of the original metrics, loses information and cannot be agnostic. Fourth, weights cannot, in general, be used to achieve definite effects.

In the next section, we present a different way of combining similarity metrics. By using return types that are partially-ordered sets of pairs, it requires no normalisation, and it does not lose any information. And, by using indifference relations, it allows customers to specify trade-offs between degrees of similarity.

## 7 Combining Similarity Metrics: Generalised Prioritisation

Our similarity metric framework allows similarity metrics to have as their return types any partially-ordered sets. In particular, we can use *pairs of values* to denote degrees of similarity, i.e. the return type of a similarity metric could just as well be a partially-ordered set of pairs as it could be a set of numbers or Plaza’s first-order feature structures.

This proves especially useful when combining component similarity metrics into composite similarity metrics. For example, suppose cases contain a price attribute and a size attribute. Suppose the probe specifies a price of £600 and a size of 100. Consider case 1 that has a price of £550 and a size of 80. Using negated absolute difference, the similarity of case 1 to the probe on price is  $-50$  and on size is  $-20$ . By the conventional approach, to form the composite degree of similarity, the two similarity values would be normalised, and the values would be averaged. But, in our approach, where we exploit the ability to have any partially-ordered set as the return type of a similarity metric, the composite metric returns pairs of values. In this approach, the similarity of case 1 to the probe is  $\langle -50, -20 \rangle$ : the pair *is* the degree of similarity. There is no normalisation; there is no combining into a single numeric value; so there is no loss of information.<sup>4</sup> If case 2 has a price of £640 and a size of 130, its similarity to the probe is the pair  $\langle -40, -30 \rangle$ .

Of course, this is not enough. Had we averaged the similarity values we would be able to see which of case 1 and case 2 was the more similar to the probe. If we use pairs to denote degrees of similarity, we similarly need to be able to compare the pairs. This is why the return type of a similarity metric has an ordering. This ordering on the pairs can be any ordering you wish to define, but most usually it will be defined from the orderings of the original component similarity metrics.

In the next few subsections, we show three general ways in which customers can form this ordering. Each is meaningful. We will take the return type of the first composite similarity metric to be the partially-ordered set  $(A, \sqsubseteq_A)$  and we will take the return type of the other metric to be  $(B, \sqsubseteq_B)$ .

## 7.1 Product

Suppose the customer wishes the first metric to contribute neither more nor less than the second. To achieve this, the customer would define an ordering on the pairs using an operator that we call *product*:

$$\langle x_1, x_2 \rangle \sqsubseteq_{A \times B} \langle y_1, y_2 \rangle \hat{=} x_1 \sqsubseteq_A y_1 \wedge x_2 \sqsubseteq_B y_2$$

i.e. for one pair to be lower in the ordering than another, both its components must be lower in their respective orderings.

It is possible, of course, for a case to be better on one component and worse on the other. The definition of product says that these pairs are incomparable, and they are thus, in some sense, ‘as good as’ each other. In the absence of any information about how we might ‘trade-off’  $A$  against  $B$ , exploiting the potential in a partial order for incomparability seems the right thing to do. The numerical averaging approach can never be agnostic in this way.

<sup>4</sup> Our earlier papers have pointed out that another advantage of this approach is that we can combine similarity metrics that have quite different return types (see, e.g., [4]). For example, a similarity metric whose return type is numeric could be combined with one of Plaza’s feature-structure-valued metrics *without inter-conversion*. This is something the averaging approach cannot do.

## 7.2 Strict Prioritisation

Consider now the case of a customer who regards one of the attributes as primary and the other as secondary. Specifically, the secondary requirement is to be met as much as possible but only after the primary requirement has been met as much as possible [6]. (The case where the primary attribute is more important but not *absolutely* more important will be dealt with in the next subsection.)

All that the customer needs here is a new ordering on the similarity pairs. We call the operator that forms this ordering *strict prioritisation*:

$$\langle x_1, x_2 \rangle \sqsubseteq_{A \gg B} \langle y_1, y_2 \rangle \hat{=} x_1 \sqsubseteq_A y_1 \vee (x_1 = y_1 \wedge x_2 \sqsubseteq_B y_2)$$

The overall ordering is based on the primary ordering ( $\sqsubseteq_A$ ) but the secondary ordering ( $\sqsubseteq_B$ ) may resolve ties in the first ordering.

Weighted averages can make one attribute more important than another, but they cannot so easily make one attribute *absolutely* more important than another. Consider, for example, combining three metrics, where the first is absolutely more important than the second and the second is absolutely more important than the third. Choosing a weight for the second metric is hard: it has to be large enough to guarantee that the second metric will be absolutely more important than the third metric, but small enough to guarantee that it will be absolutely less important than the first metric. Particular weighting schemes can be crafted for particular metrics. But there is no *general* approach using weighting, and this is the value of strict prioritisation.

## 7.3 Generalised Prioritisation

Customers have another way of combining partial-orders, which we call *generalised prioritisation*:

$$\langle x_1, x_2 \rangle \sqsubseteq_{A \gg \simeq B} \langle y_1, y_1 \rangle \hat{=} x_1 \sqsubseteq_A y_1 \wedge (x_1 \simeq_A y_1 \Rightarrow x_2 \sqsubseteq_B y_2)$$

As you can see, this definition offers another use for indifference relations,  $\simeq$ . It gives us an ordering which, to some extent, favours its first argument, but not necessarily to the extent of strict prioritisation. We order the pairs according to the first value in the pair, unless we are indifferent between the first values, in which case we take both components of the pair into account.

This is the second distinct use of indifference in this paper. As before, we claim that it satisfies the Requirements and Results Criteria quite well. We hope that customers will find indifference meaningful during specification of requirements. And, they should be able to interpret the results they see in terms of their indifference relation: there can be worse matches on the first metric (worse, but only up to the customer's level of indifference) if they are compensated for by better matches on the second metric.

In passing, we should point out that generalised prioritisation has the nice property that it has product and strict prioritisation as special cases. If we prioritise order  $A$  over order  $B$  but we define  $x \simeq_A y = \mathbf{True}$ , then we obtain

an ordering identical to  $\sqsubseteq_{A \times B}$ , the product of the two orders. If we prioritise order  $A$  over order  $B$  but we use equality for  $\simeq_A$ , then, after simplification, the definition reduces to our previous definition for strict prioritisation,  $\sqsubseteq_{A \gg B}$ .

## 8 Conclusion

We have argued that conventional case retrieval technology exhibits a level of unjustified precision which can complicate specification of customer requirements and interpretation of retrieval results in terms of those requirements.

We have proposed an approach that uses meaningful, partially-ordered return types for similarity metrics (especially for composite metrics) and a notion of ‘indifference’ between degrees of similarity. By accommodating partial-orders, customers are not obliged to use over-specified metrics. And by using indifference relations, we give explicit control over the precision of metric return types.

We have built a case retrieval system that works in the way described in this paper. To the user, the system looks much like conventional case-retrieval systems. The knowledge engineer will have supplied the individual similarity metrics. A choice of combining operators (section 7) must be made and definitions of indifference must be given (which can sometimes be as simple as specifying a value for  $\epsilon$ ). The results of a query again look much as they do in other systems. The difference is that the result set need not be of some arbitrary predetermined size (although, since our framework subsumes conventional approaches, customers can issue ‘best- $n$ ’ queries if they really wish to). The system offers a rich set of options for query revision [3].

## References

1. Bergmann, R., Breen, S., Göker, M., Manago, M. & Wess, S.: *Developing Industrial CBR Applications*, LNAI-1612, Springer, 1999 74, 76
2. Bridge, D. G.: Defining and Combining Symmetric and Asymmetric Similarity Measures, in B.Smyth & P.Cunningham (eds.), *Advances in CBR (Procs. of 4th European Workshop on CBR)*, LNAI-1488, pp.52-63, Springer, 1998 75
3. Ferguson, A. & Bridge, D.: Options for Query Revision when Interacting with Case Retrieval Systems, in I.Watson (ed.), *Procs. of the Fourth UK CBR Workshop*, University of Salford, 1999 75, 85
4. Osborne, H. & Bridge, D.: Similarity Metrics: A Formal Unification of Cardinal and Non-Cardinal Similarity Measures, in D. B.Leake and E.Plaza (eds.), *CBR Research and Development (Procs. of Second International Conference on CBR)*, LNAI-1266, pp.235-244, Springer, 1997 75, 83
5. Plaza, E.: Cases as terms: A feature term approach to the structured representation of cases, in M.Veloso & A.Aamodt (eds.), *CBR Research and Development (Procs. of First International Conference on CBR)*, LNAI-1010, pp.265-276, Springer, 1995 75
6. Vollrath, I.: Handling Vague and Qualitative Criteria in Case-Based Reasoning Applications, in the on-line pre-proceedings of the Eighth German Workshop on CBR, 2000 (<http://wwwagr.informatik.uni-kl.de/~gwcbr2k/program.html>) 84
7. Wilson, D. R. & Martinez, T. R.: Improved Heterogeneous Distance Functions, *Journal of Artificial Intelligence Research*, vol.6, pp.1-34, 1997 80

# Representing Knowledge for Case-Based Reasoning: The ROCADE System

Béatrice Fuchs<sup>1</sup> and Alain Mille<sup>2</sup>

<sup>1</sup> Université Lyon III, IAE-Modeme  
15 quai Claude Bernard, 69 007 Lyon, France

<sup>2</sup> Université Lyon I, LISI  
Bât. 710, 43 bd du 11 novembre 1918, 69 100 Villeurbanne  
fuchs@univ-lyon3.fr amille@bat710.univ-lyon1.fr

**Abstract.** This paper presents the object-based knowledge representation system ROCADE, that is aimed at the development of case-based reasoning (CBR) systems. CBR is studied by reference to the two levels defined by Newell: at the knowledge level, a general detailed model of the CBR process has been proposed. This model is intended to be implemented at the symbol level materialized by the ROCADE system. This paper presents these two complementary levels and focuses on ROCADE. The concepts and reasoning mechanisms of ROCADE are described, as well as its architecture. Then, its architecture allowing different ways to use it is presented. ROCADE is illustrated with examples of two CBR systems. The implementation of 2 CBR systems are used to illustrate the rocade system the functionalities of the rocade system

## 1 Introduction

The case-based reasoning (CBR) research field has reached a good maturity and a lot of systems have been developed for various tasks and domains. Moreover commercial CBR shells implementing some CBR functionalities have been developed. It appeared to be necessary to study more rationally the development process of CBR systems. With respect to the two levels defined by Newell [Newell, 1982], this study has been carried out with respect to by considering the two aspects, the knowledge level and the symbol level. At the knowledge level, we proposed a framework for the analysis of CBR systems, that has been used in [Fuchs and Mille, 1999] for the study of the adaptation process. In this paper, we propose to present the knowledge representation system ROCADE that has been implemented in response to CBR systems development needs, especially for the development of the Pad'im and Designer systems [Fuchs et al., 1995] in the domain of industrial supervision.

The paper is organized as follow: section 2 introduces the problem of CBR system design and development, and summarizes the principles of the two level approach that we chosen. Section 3 focuses on the knowledge level and presents the modeling principles that we retained in our framework. It is followed by the

section 4 that focuses on the symbol level and describes the features of ROCADE by contrast with other CBR shells and knowledge representation systems. Section 5 discusses advantages and drawbacks of ROCADE and is followed by a conclusion.

## 2 Principles of the Approach

The development of CBR systems as well as any knowledge-based system (KBS) requires the acquisition of a big amount of knowledge that is then translated into a knowledge representation system (KRS). KRS implement representation primitives in order to express knowledge. These primitives are associated to transformation mechanisms that allow to produce new knowledge responding to questions that are asked to the system. KBS development implies two important phases [Aamodt, 1991]. The first one is the initial acquisition phase, sometimes associated to a modeling phase that aims at collecting all the required knowledge: domain knowledge, problem solving knowledge, *etc.* The second one is the maintenance phase of existing knowledge that aims at enhancing existing knowledge in order to correct possible errors introduced in the initial modeling phase, evolve in conformity with the real world, enhance the system's efficiency, *etc.* Furthermore, Newell distinguished two levels of knowledge [Newell, 1982]: the knowledge level whose focus is more abstract and independent of the implementation and the symbol level that corresponds to the implementation. These two levels have been studied in a number of previous works on the development of CBR systems.

From the knowledge level point of view, some generic conceptual models of CBR have been proposed [Slade, 1991], or [Aamodt and Plaza, 1994], which are most of the time very general and are not sufficiently detailed to constitute a real guide of low-level reasoning tasks. Some more detailed models have been proposed, such as [Armengol and Plaza, 1994] or [Goel, 1996], but for specific application domains and tasks. More in a software engineering trend, the INRECA methodology aims at providing guidelines to support the development of CBR applications [Bergmann and Althoff, 1998]. However no generic model is used to support and guide both design and implementation. At the knowledge level, there is a lack of a precise and detailed model that would facilitate the analysis and the design of CBR systems.

At the symbol level, there is two categories of environments. The first category is those of commercial CBR shells that implement some of the CBR functionalities [Althoff et al., 1995a], [Lenz et al., 1998] in a generic way and propose efficient retrieval strategies. The main drawback of these tools is that the adaptation and learning steps are most of the time not taken into account in the development process, or considered as separate steps of the application. Furthermore, there is no explicit model of the reasoning process and the implicit underlying model does not give to a designer a sufficient degree of freedom to customize a specific application. Consequently, the mechanisms proposed in these tools are most of the time predefined and designed

for particular application categories (e.g. diagnosis, decision support). The development of a particular application needing some specific features is difficult because these tools are not easily adaptable. The second category is those of KRS such as Creekl [Aamodt, 1991], Noos [Plaza and Arcos, 1993], or Casuel [Althoff et al., 1995b] for example. These systems are general purpose systems and do not provide a model for guiding the design and implementation of CBR systems. Another aspect that is rarely handled in development environments is the ability to communicate with databases in order to collect information that can help constituting cases. When designing CBR systems for industrial supervision [Fuchs et al., 1995], we had to include these functions in the reasoning models and to provide specific mechanisms so that the CBR system is able to collect the needed information from the external information system.

In our approach, instead of considering separately the definition of a case, of the cases organization, *etc.* our approach tends to present all CBR aspects as a knowledge representation problem since it covers most of the needs for the implementation of CBR systems [Kamp et al., 1998] and is open enough to allow the implementation of particular features for specific applications. The second important characteristic of our approach is that the knowledge and symbol levels are connected so that all the models that have been analysed at the knowledge level are translated to the symbol level. These models may be then specialized for a particular application. This enhances the trace and the history of the development process. A consequence to this is that the learning phase is not yet limited to the storage of a case or the organization of the case base, but may involve a reasoning step that is enabled thanks to the explicitation of knowledge.

### 3 The Knowledge Level

#### 3.1 CBR Modeling

CBR is a complex process where every step can be related to different areas in artificial intelligence. This probably explains the lack of modeling and the difficulty to propose generic models. In this domain the research community agrees on the general principles, but there is no consensus on a precise definition of the case notion, and the exact content of every reasoning step is not clear. In particular the adaptation step is the most important of the CBR cycle, and the most difficult to handle. Among research works devoted to the CBR process modeling, we can consider two categories.

The first category is those of the *knowledge engineering* field in general, whose goal is to propose knowledge acquisition methodologies for designing knowledge-based systems. The KADS methodology [Wielinga et al., 1992] is an example, but at our knowledge, no convincing use has been performed since the combination of the retrieval-adaptation steps becomes complex, for example systems performing adaptation-guided retrieval. We can cite also the *components of expertise* methodology [Steels, 1990] that has inspired several modelings of CBR SYSTEMS. However, these analysis have been performed specifically to a restricted domain. These methodologies rely on the same basic principles, i.e.



they distinguish several knowledge categories (tasks, methods, inferences, domain) ; they specify the role played by knowledge in the reasoning process, and they advocate the reusability of knowledge for other systems. In practice, the knowledge-level analysis is rarely implemented in an environment that enables easily the transposition of the concepts introduced at this level in a development environment. This is a considerable limitation of the reuse capabilities. Furthermore, there is a lot of generic low-level tasks (e. g. the matching process), but there are many ways to connect them. Existing task models do not allow to describe entire generic task structures that would be specializable.

The second category is devoted to the modeling of CBR in general. Several modelings using the reasoning task concept have been proposed: [Goel, 1996] presents a framework for describing the case-based design system KRITIK, [Armengol and Plaza, 1994] analyses the CHEF system by applying the *components of expertise* methodology, and [Aamodt and Plaza, 1994] proposes an ontology of the main CBR tasks. These models are either too specific to an application domain of a particular system or are too general to constitute a low-level description support of reasoning tasks. Meanwhile, several interesting principles of these works have inspired our approach. First, we have considered both knowledge and symbol levels. In this perspective, our knowledge modeling approach rely on the principles of COMPONENTS OF EXPERTISE [Steels, 1990] because the method requires a connexion between the two knowledge and symbol levels. We defined a task formalism inspired from [Aamodt and Plaza, 1994] on which we added some specific annotations specifying the specialization, ordering and multiple decomposition of tasks. Finally the meta-case notion of [Goel, 1996] and the duality between case and reasoning tasks is the foundation of our model. The main principles of our approach are presented in the next section.

### 3.2 Modeling Principles of the Framework

In the modeling framework that has been designed for CBR, knowledge is splitted into a set of *knowledge models*. A knowledge model associates concept descriptions that are classified depending on the role they play in the problem solving process. Thus, cases are described in a *case model*, indexing is described in an *indexing model*, etc. We defined three main categories of models.

The *conceptual domain model* introduces the concepts used to describe the domain ontology independently of the reasoning process. It constitutes the basic "kernel" i. e. the common vocabulary to which all other models make reference. In the same way, the *case model* describes the experiences and in particular the problem part, the solution and the process that leads from problem to solution: the reasoning trace.

The *reasoning tasks models* describe the overall CBR process: elaboration, retrieval, adaptation, etc. by using two formalisms: a task decomposition formalism and a task specification fomalism. The task decomposition formalism outlines the decomposition of the main CBR task into several subtasks recursively. The task specification fomalism describes the tasks individually. A task is described by the pieces of knowledge it processes in input, a label clarifying



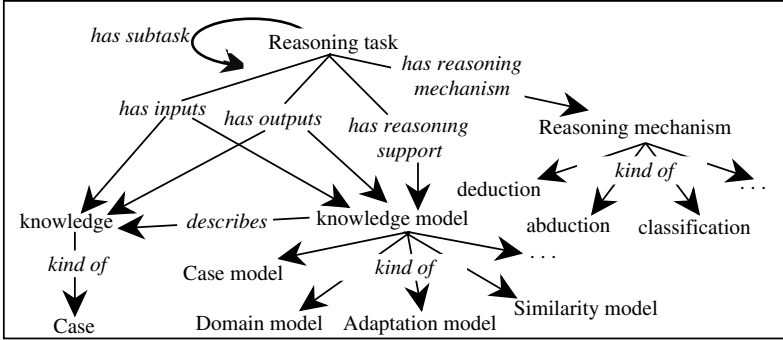


**Fig. 1.** A task oriented view of the concepts involved in the framework

the functions performed by the task, the pieces of knowledge it produces as output, the knowledge models used as support, and the reasoning mechanisms implemented (fig. 1).

The *reasoning support models* precise the needed knowledge to the achievement of every task: how to index cases, assess similarity, adapt a solution, *etc.*

An important principle of the framework is that there is a duality between the case and the reasoning task: the case contains the reasoning trace leading from a problem to a solution as well as the support knowledge necessary to achieve it ; and the reasoning task model describes the progressive transformation of a case through the different steps until its storage in the case base.

The objective of the modeling is firstly to make explicit the knowledge used and produced by the reasoning tasks: this is the role of the *task specification model*, and secondly to precise the interconnections of the different tasks for the whole problem solving: this the role of the *task decomposition formalism*. The principle is to structure the reasoning process into hierarchical tasks with an associated default control, and to define precisely each task by relating it to the different knowledge models and reasoning mechanisms that are implied to achieve it.

The analysis of the whole CBR process for several applications in different domains and cognitive tasks has led us to generalize these approaches and to build a generic detailed model of the CBR process [Fuchs, 1997]. This generic model has been able to underline the kind of pieces of knowledge processed by the reasoning process, the kind of inference mechanisms controlling the reasoning process: classification, matching, *etc.*, and the knowledge models needed to support the reasoning process: specification of the similarity used, indexing criteria, adaptation rules, *etc.*

## 4 The Symbol Level

### 4.1 Knowledge and Inference Needs

The specification task model of CBR outlines the different knowledge models that are used and the reasoning mechanisms that support the CBR process.

A development environment for CBR systems must provide primitive entities to represent domain knowledge: concepts and their properties, the taxonomy of concepts, relations between concepts (composition, functional relations expressing the role played by an element in another one, causal relations, *etc.*). In the same way cases are represented using the concepts of a problem, a solution and a reasoning process. A problem and its solution are described using the vocabulary defined in the conceptual domain model. As described previously, a concept may be viewed independently of its use (conceptual models) and also through the reasoning process (support models). This corresponds to the notion of point of view: this notion has been introduced in order to describe independently several aspects of the same concept. A point of view expresses a position where an object may be observed [Mariño et al., 1990]. A concept is unique but there are several manners to describe it. The notion of point of view is useful to split the knowledge definition into several complementary models because it allows to define knowledge in a modular manner.

For supporting reasoning capabilities, a symbol level environment must provide inference mechanisms whose combination performs the required reasoning capabilities. For domain knowledge, an inheritance mechanism allows to implement a taxonomy of domain concepts and to deduce new properties according to the position of a concept in an existing conceptual hierarchy. For the elaboration of a case before retrieval, it should be possible to access to external information source in order to collect cases description, and further to infer new descriptors from previous ones. The elaboration step often builds an index by an abstraction process. This index is intended to be used in the retrieval step. For the retrieval of cases, a matching process is needed that compares concepts and their properties in order to establish relations between them that express their similarity. Classification mechanisms are often used in order to organize the case base according to significant characteristics and to retrieve a set of cases. Categorization is another way for grouping individual concepts according to their common properties, although they may have significant differences. A generalization process that produces a unique concept description from a set of examples may be used. For the similarity justification as well as for learning, the generation of explanations is often used to support and justify plausible inferences, or to diagnose some problems, *etc.* For the adaptation step, some deductive mechanisms are often used, allowing to trigger and chain the execution of rules for example for performing basic adaptation operations. Constraints satisfaction is a mechanism that is often used in the adaptation process, and it relies on the representation either of local constraints on objects, or on constraints networks that are more global because they involve several concepts. For storing a case in the case base, classification mechanisms are needed to manage a case hierarchy.

Many techniques may be used to build and to update a hierarchy: a subsumption hierarchy [Napoli et al., 1996], discrimination or induction tree, *etc.* A flexible tool must be able to implement any alternative technique.

For the symbol level, we have retained the object-based representation. The reasons for this are that it allows to represent easier structured knowledge, it provides basic inference methods whose combination allows a variety of reasoning methods [Fikes and Kehler, 1985], and the concepts involved are more closed to the real world concepts.

## 4.2 Symbol Level Development Tools for CBR Systems

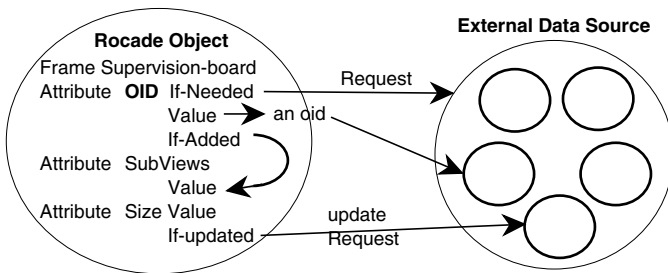
Some object-based knowledge representation systems have been implemented such as the Creekl language [Aamodt, 1991], or the Noos system [Plaza and Arcos, 1993], most of them are Lisp-based languages, and do not co-operate interactively with their environments. Most commercial tools are software engineering environments that provide predefined CBR functionalities, for example the CBR-Works system or CBR\*TOOLS [Jaczynski and Trousse, 1998]. The needs of our applications and more generally for systems that are opened on the information system of the organization has led us to consider other kinds of tools oriented on knowledge representation. The ROCADE system, discussed in the next section, is an object-based knowledge representation system designed for the development of CBR systems.

## 4.3 General Presentation of ROCADE

ROCADE is an object-based knowledge representation system. It has been implemented as a package of Objective-C classes that are instantiated in order to create the objects of a knowledge base. Classes contain a set of methods for the manipulation of knowledge in order to perform basic inference mechanisms. In ROCADE, knowledge is described with triples (frame, attribute, facet). A ROCADE knowledge base contains three kinds of components: **frames** that are used to describe domain knowledge, **cases** for case-specific knowledge and **slots** that define properties of the slots (names, inverse relations, *etc.*) The different facets that may be used are the range facet, the default value and value facets, constraint facets and procedural facets. Constraint facets include the cardinality facet, interval facet and value class facet. In the value class facet, the allowed values for the attribute must have a specialization relation with an object of the set defined in the value class facet. Procedural facets contain procedures which are triggered when attributes values are accessed. They are written using the Objective-C language, compiled and dynamically loaded at execution. The different procedural facets are **If-removed**, **If-added** and **If-needed**. Objects can be organized in a multiple inheritance hierarchy that allows specific objects (subclasses) to inherit knowledge from superclasses. Some consistency verification mechanisms are triggered recursively at every modification in the hierarchy and the final value of an attribute is determined dynamically when the attribute is accessed. The access to an attribute triggers a complex process because of

the existence of several kinds of facets for the determination of the value, and because of the inheritance of values in the hierarchy. When reading the value of an attribute, the procedural facet If-Needed is first examined. Its execution returns a value that is set in the value facet of the attribute, and classical search methods may then be used in order to read this value. When setting or modifying an attribute value, procedural facets are triggered. After a verification of the consistency of the modification, the updating of the value is performed and the procedural facets If-added, if-remove corresponding to the actions performed are triggered. The modification is done locally, since the inherited values are not copied, but the triggering of facets is propagated in the hierarchy. Several classification mechanisms have been experienced. The first one corresponds to a dynamic memory model. An object is classified depending on the object that are already present in the hierarchy. Another kind of classification allows to specify a subset of attributes that have only to be taken into account for building a hierarchy. This subset is modifiable and the hierarchy is re-built dynamically.

In ROCADE, a mechanism has been implemented in order to collect information from external sources and to constitute knowledge and cases. It was initially set for the needs of an industrial supervision application, the Pad'im system, [Fuchs et al., 1995], in order to collect supervision data and to constitute the context of a situation. It turned out that this mechanism was useful for collecting other knowledge from various data sources (databases). The principle was to create a set of ROCADE objects containing procedural facets that query external data sources and update in response attributes of the ROCADE objects. In the same way, the modification of attribute values of the ROCADE objects triggers update requests towards external data sources. The following schema summarizes this mechanism.



This mechanism has been used in order to collect the overall structure of a supervision system (supervision board, multimedia views, etc) into the ROCADE knowledge base.

#### 4.4 Architecture of ROCADE

The architecture of ROCADE has been designed in order to enable a high degree of freedom and a high level of flexibility when implementing a CBR system. The architecture is modular in order to have a high reusability. ROCADE features two subsets of classes: the engine and the default user interface (fig. 2).

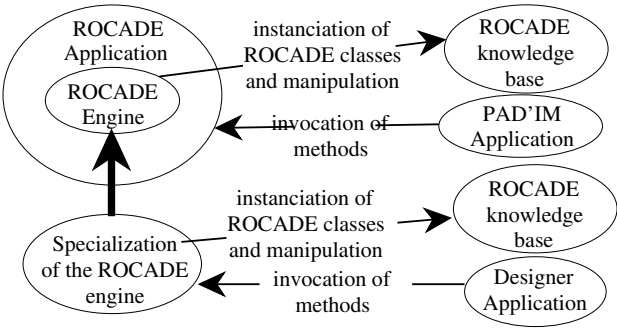


**Fig. 2.** The architecture of ROCADE and different ways to use its features

The ROCADE engine is a set of classes that are independent of a particular application and of the graphical interface. The classes of the ROCADE engine are used to create the objects of a knowledge base. For example, there is a class named *Frame* whose instances are objects of the knowledge base. There is an Objective-C class for every concept of the system (attributes, value facet, procedural facet, *etc.*). The classes of the engine are defined as usual Objective-C classes: they have instance variables and associated methods (class and instance methods). The classes of the engine can be instanciated and the objects are used in specific applications by invoking methods that are implemented in classes (classes are also objects).

The interface is composed of interface classes that are associated to objects of the ROCADE engine, and allow to access in a more easy manner to objects of the knowledge base. A browser (fig. 3) allows the navigation within the network of objects and relations and allows basic operations. This interface constitutes the default interface of ROCADE allowing to visualize objects of the knowledge

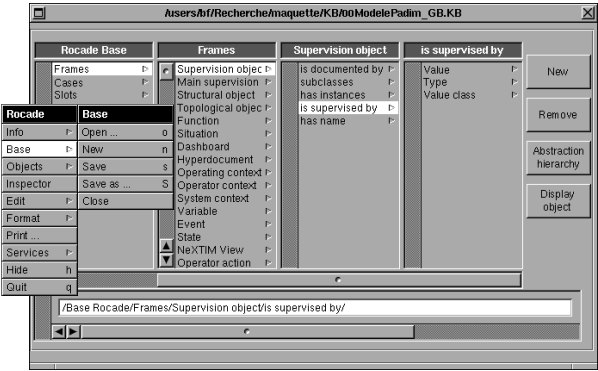


**Fig. 3.** The basic interface of ROCADE

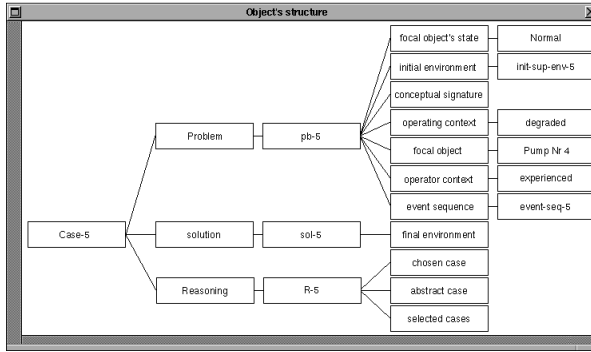


**Fig. 4.** The graphic visualization of a case

base, and to propose basic actions on these objects (add, remove, display). When selecting an object in the browser, a specific interface to this kind of object, an *inspector*, is displayed and allows more accurate operations.

Some visualization functions allow to display either a graphical representation or a textual representation. The graphical visualization represents either an inheritance hierarchy or a case hierarchy, or the network of relations between objects (fig. 4).

The classes of the ROCADE engine may be specialized by using the provided inheritance mechanism of Objective-C (fig. 2). The inference mechanisms of ROCADE are implemented using various Objective-C methods. These methods may be overridden by using the overloading mechanisms of Objective-C. The case-based designer system *Designer* illustrates this kind of use of the ROCADE objects : higher-level methods have been defined (a layer above the engine) that facilitate the manipulation of the knowledge base. The user interface of ROCADE have not been used because too general, and a new interface has been build for the specific domain objects. Using ROCADE, two applications have been developed with the Rocade system: Pad'im and Designer. Pad'im is a decision support system in the domain of industrial supervision. It has been implemented using the ROCADE engine and the interface, without adding new features nor interfaces. Designer is a Case-based Design system in the domain of industrial supervision. Its implementation is based on ROCADE but in different way. The basic methods of the ROCADE engine have been used in order to build higher-level methods specifically for the needs of the application. The basic ROCADE user interface is too generic and has not been reused for this application, so a more specialized user interface has been developed.

## 5 Discussion

The ROCADE system is a first prototype and thanks to these two first implementations, we can enlight some limitations and benefits.

Attributes are first-class citizen i.e. classes representing attributes are defined and named attribute classes. ROCADE uses attribute classes in order to define common properties of attributes (type, inverse relation, name, *etc.*), but they should be more elaborated than it is presently. The advantage of attribute classes is to attach a global behaviour to a class of attributes and to inherit this behaviour everywhere the attribute is defined and used. The basic methods of ROCADE for manipulating objects, attributes and values are low-level methods, and consequently using the functionalities of ROCADE is sometimes tedious. When implementing the Designer system, the developer began developing higher-level manipulation methods. However, this drawback underlines another benefit of ROCADE: the ability to specialize and reuse easily the set of existing methods. Another drawback of ROCADE is that it does not provide a sufficient set of basic objects (only basic attributes for inheritance and instantiation are predefined). This limitation is not directly due to ROCADE. It should be rather considered as a further extension to develop and ROCADE could take advantage of a collection of predefined objects.

A perspective to this work is to implement the knowledge-level generic models using ROCADE in order to facilitate the design and implementation of a particular CBR system by specializing and instantiating the generic models. So ROCADE would be not only a development system, but also an environment for analyzing, designing and implementing CBR systems. The knowledge-level concepts could then be reused in the implementation step and this would contribute to the integration of the two complementary knowledge and symbol levels.

## 6 Conclusion

In this paper, we have presented the ROCADE system, an object-based knowledge representation system dedicated to the development of CBR systems. This system provides representation primitives to represent domain knowledge and cases components, basic inference methods whose combination provide the required reasoning capabilities of CBR systems, and communication mechanisms to send request to databases of the organization. The architecture is open and there are many ways to use the system: use the default methods or override them, use the basic default interface or build a new one. Two implementations of CBR systems have helped to analyze and evaluate the capabilities of ROCADE.

## References

- Aamodt, 1991. Aamodt, A. (1991). *A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning*. Doctoral dissertation, Trondheim University, Trondheim, Norway. [87](#), [88](#), [92](#)
- Aamodt and Plaza, 1994. Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–58. [87](#), [89](#)
- Althoff et al., 1995a. Althoff, K.-D., Auriol, E., Barletta, R., and Manago, M. (1995a). *A Review of Industrial Case-Based Reasoning Tools*. *AI Intelligence*. [87](#)

- Althoff et al., 1995b. Althoff, K.-D., Auriol, E., Bergmann, R., Breen, S., Dittrich, S., Johnston, R., Manago, M., Traphoner, R., and Wess, S. (1995b). Case-Based Reasoning for Decision Support and Diagnostic Problem Solving: The INRECA Approach. In *Proc. of the 3rd Workshop of the German special Interest Group on CBR at the 3rd German Expert System Conference*. 88
- Armengol and Plaza, 1994. Armengol, E. and Plaza, E. (1994). A Knowledge Level Model of Case-Based Reasoning. In Richter, M. M., Wess, S., Althoff, K.-D., and Maurer, F., editors, *First European Workshop on Case-Based Reasoning - EWCBR-93*, pages 53–64, Kaiserslautern, Germany. LNAI, vol. 837, Springer, Berlin. 87, 89
- Bergmann and Althoff, 1998. Bergmann, R. and Althoff, K.-D. (1998). *Methodology for Building CBR Applications*, chapter 12, pages 299–326. In [Lenz et al., 1998], LNAI 1400. Springer. 87
- Fikes and Kehler, 1985. Fikes, R. and Kehler, T. (1985). The Role of Frame-Based Representation in Reasoning. *Communications of ACM*, 28(9):904–920. 92
- Fuchs, 1997. Fuchs, B. (1997). *Représentation des connaissances pour le raisonnement à partir de cas : le système ROCADE*. Thèse d’université, Université Jean Monnet, Saint-Etienne, France. 90
- Fuchs and Mille, 1999. Fuchs, B. and Mille, A. (1999). A Knowledge-Level Task Model of Adaptation in Case-Based Reasoning. In Branting, K., Althoff, K.-D., and Bergmann, R., editors, *Third International Conference on Case-Based Reasoning - ICCBR-99*, pages 118–131, Seeon, Germany. LNAI, Springer, Berlin. 86
- Fuchs et al., 1995. Fuchs, B., Mille, A., and Chiron, B. (1995). Operator Decision aiding by Adaptation of Supervision Strategies. In Veloso, M. and Aamodt, A., editors, *First International Conference on Case-Based Reasoning - ICCBR-95*, pages 23–32, Sesimbra, Portugal. LNAI, vol. 1010, Springer, Berlin. 86, 88, 93
- Goel, 1996. Goel, A. (1996). Meta cases: Explaining case-based reasoning. In Smith, I. and Faltings, B., editors, *Third European Workshop on Case-Based Reasoning - EWCBR-96*, pages 150–163, Lausanne, Suisse. LNAI, vol. 1168, Springer, Berlin. 87, 89
- Jaczinski and Trousse, 1998. Jaczinski, M. and Trousse, B. (1998). An object-oriented framework for the design and implementation of case-based reasoners. In *Proceedings of the 6th German Workshop on CBR (GWCBR-98)*. 92
- Kamp et al., 1998. Kamp, G., Lange, S., and Globig, C. (1998). *Related areas*, chapter 13, pages 327–351. In In [Lenz et al., 1998] LNAI 1400. Springer. 88
- Lenz et al., 1998. Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., and Wess, S. (1998). *Case-Based Reasoning Technology, from foundations to applications*. LNAI 1400. Springer, Berlin. 87, 97
- Mariño et al., 1990. Mariño, O., Rechenmann, F., and Uvietta, P. (1990). Multiple perspectives and classification mechanism in object-oriented representation. In *Proceedings of the 9th European Conference on Artificial Intelligence - ECAI’90*, pages 425–430, Stockholm (SE). Pitman Publishing, London (GB). 91
- Napoli et al., 1996. Napoli, A., Lieber, J., and Curien, R. (1996). Classification-Based Problem-solving in Case-Based Reasoning. In Smith, I. and Faltings, B., editors, *Third European Workshop on Case-Based Reasoning - EWCBR-96*, pages 295–308, Lausanne, Suisse. LNAI, vol. 1168, Springer, Berlin. 92
- Newell, 1982. Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 19(2):87–127. 86, 87
- Plaza and Arcos, 1993. Plaza, E. and Arcos, J.-L. (1993). Noos: an integrated framework for problem solving and learning. Technical report, Institut d’investigació en Intelligència Artificial, Barcelona, Spain, Report IIIA-RR-97-02. 88, 92



- Slade, 1991. Slade, S. (1991). Case-Based Reasoning: A Research Paradigm. *AI Magazine*, 12(1):42–55. [87](#)
- Steels, 1990. Steels, L. (Summer 1990). Components of Expertise. *AI Magazine*, pages 28–49. [88](#), [89](#)
- Wielinga et al., 1992. Wielinga, B. J., Schreiber, T., and Breuker, J. A. (1992). KADS: A Modelling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4(1):136–145. [88](#)

# Personalized Conversational Case-Based Recommendation

Mehmet H. Gker<sup>1</sup> and Cynthia A. Thompson<sup>2</sup>

<sup>1</sup>DaimlerChrysler Research & Technology  
1510 Page Mill Road, Palo Alto, CA 94304  
mehmet.goeker@daimlerchrysler.com  
<sup>2</sup>Center for the Study of Language and Information  
Stanford University, Stanford, CA 94305-4115  
cthomp@csl.i.stanford.edu

**Abstract:** In this paper, we describe the Adaptive Place Advisor, a user adaptive, conversational recommendation system designed to help users decide on a destination, specifically a restaurant. We view the selection of destinations as an interactive, conversational process, with the advisory system inquiring about desired item characteristics and the human responding. The user model, which contains preferences regarding items, attributes, values, value combinations, and diversification, is also acquired during the conversation. The system enhances the user's requirements with the user model and retrieves suitable items from a case-base. If the number of items found by the system is unsuitable (too high, too low) the next attribute to be constrained or relaxed is selected based on the information gain associated with the attributes. We also describe the current status of the system and future work.

## 1 Motivation

As information becomes abundant, humans are confronted with more difficult decisions about how to access, navigate through, and select available options. The sheer number of alternatives often makes a wise choice impossible without some intelligent computational assistance. In response to this need, there have been increased efforts to design and implement intelligent aides for filtering web sites (e.g., Pazzani, Muramatsu, & Billsus (1996)), news stories (e.g., Lang (1995)), TV listings (Smyth and Cotter, (1999)), and other information sources. A related line of research and development has led to *recommendation systems* (e.g. Burke, Hammond, and Young (1996), Resnick and Varian (1997), Burke (1999)), which can be used for any task that requires choice among a large set of predefined items.

Society, on the other hand, is getting more complex and diversified. The differences in personal preferences, social and educational backgrounds, and private or professional interests are increasing, and tools to access information are becoming

ubiquitous. This causes the need for intelligent systems that process, filter, and display available information in a personalized manner. Research on personalization has led to the development of systems that adapt themselves to the characteristics of their user: *user adaptive systems* (c.f. Rich, E. (1979), Langley (1997)).

In this paper we describe the Adaptive Place Advisor, a user adaptive, conversational recommendation system. The system helps the user to select a destination, for example a restaurant, from its database by performing a personalized conversation.

## 2 Conversational Recommendation Systems

Given a large set of items and a description of the user's needs, recommendation systems present to the user a small set of the items that are suited to these requirements.

The most widely used framework for recommendation systems is the *ranked list* approach. In this scheme, the user specifies his needs with one or more keywords and the system presents a usually long list of results, ordered by their predicted relevance to the user. This technique comes from information retrieval, where it was originally designed to help find documents or reports during library research. However, the basic method is quite general, and it underlies most search engines for the World Wide Web, which millions now use regularly for many quite different selection tasks.

Yet despite its current popularity, the ranked list scheme is not the only approach to making recommendations. Consider a situation in which one person, that we will call the *inquirer*, asks a second person, that we will call the *advisor*, for assistance in deciding on a restaurant at which to have dinner:

- Inquirer:* Where do you think I should eat tonight?  
*Advisor:* Well, what type of cuisine would you like?  
*Inquirer:* What types are there?  
*Advisor:* Some examples are Chinese, Indian, and Mediterranean.  
*Inquirer:* Oh, maybe Indian.  
*Advisor:* What quality rating would you like?  
*Inquirer:* I don't care, as long as it's cheap.  
*Advisor:* How do you want to pay?  
*Inquirer:* No, I think I'd like Chinese instead.  
*Advisor:* Okay, we'll switch to Chinese food. What city do you prefer?  
*Inquirer:* How about Berkeley?  
*Advisor:* I know three cheap Chinese restaurants in Berkeley. One is the Long Life Vegi House on 2129 University Avenue. Does that sound alright?  
*Inquirer:* Sure, that sounds fine.

We will refer to systems that mimic this approach to recommendation as *conversational* recommendation systems (c.f. Aha and Breslow (1997)).

The interaction supported by conversational systems seems quite different from that found in the ranked list approach. The most important distinction is that the inquirer never hears about a complete item until only one, or at most a few, choices remain. Rather than being overwhelmed with items that compete for his attention, the

user interacts with the advisor to narrow down the choices in an iterative, manageable fashion. This interaction takes the form of a sequence of questions, most designed to eliminate some items from consideration. Answering these questions plays a similar role to giving keywords with the ranked list scheme, but the aim is to remove alternatives rather than to simply order them. The conversational process can also help the inquirer better understand his own desires, since thinking about possible questions and answers may clarify goals in ways a ranked list does not.

Such dialogues seem better for recommendations that must be delivered by speech rather than visually, such as ones engaged in while the inquirer is driving. They also seem ideal, independent of modality, for tasks like destination selection or help-desk applications (c.f. Aha and Breslow (1997), Gker and Roth-Berghofer (1999)), in which the user needs to converge on at most a few items. On the other hand, ranked list methods seem more appropriate in situations where information can be presented visually and for tasks like the selection of web pages or news stories, in which the user may well want to examine many options.

### 3 User Adaptive Systems

Raw data usually does not change based on the individual processing it. However, the resulting information and the manner in which it is presented can be influenced by personal differences.

Diversification in society has a direct impact on the number of ways in which users may prefer their data to be processed, selected, and presented. A computer system should ultimately be sophisticated enough to take individual variations in preferences, goals, and backgrounds into account and generate personalized information.

User adaptive systems accommodate individual preferences by building and utilizing user models. These models can represent stereotypical users or individuals, they can be handcrafted or learned (from questionnaires, ratings, or usage traces), and they can contain information about previously selected items, preferences regarding item characteristics, or properties of the users themselves (c.f. Rich (1979)).

The individual differences represented in the user model can have an effect on computer systems at the data processing level, the information filtering level, and the information presentation level. The effects can be based on the on the content of the processed data (*content based approach*, c.f. Pazzani et.al. (1996), Lang (1995)), on how comparable data was processed by other users (*collaborative approach*, c.f. Konstan, Miller, Maltz et. al. (1997), Billsus and Pazzani (1998)), or on a mixture of both.

In summary, user adaptive systems are intelligent systems that assess user preferences and change their behavior accordingly, on one or more of the above mentioned levels.

## 4 The Adaptive Place Advisor

Our goal is to develop conversational recommendation systems in which the interaction between the system and user becomes more efficient over time due to the system’s adjustments to the preferences of the user.

In the following sections, we describe the *Adaptive Place Advisor*, a conversational recommendation system designed to help users decide on a destination. Our system adapts its behavior on the information filtering level and, by changing the order of the dialogue operators in the conversation, the information presentation level. While this approach does extend to item recommendation in general, our initial work has focused on destination selection as the application domain. Our prototype system aims to help drivers select a restaurant that meets their preferences. The system is built on a case-based paradigm and utilizes traces of the interaction with the user to adapt its similarity calculation, thereby personalizing the retrieval and the conversation.

To be able to recommend a restaurant based on a conversation, the Adaptive Place Advisor has to

- carry out a conversation and generate a partial restaurant specification, i.e. a query,
- improve or complement the query with a model of the user’s preferences,
- use this query to retrieve matching restaurants from a database and calculate their similarity to the user’s request, and
- if the number of retrieved items is not acceptable, select the next attribute to be constrained or relaxed during a conversation, and
- learn and update the user model based on these interactions.

The responsibilities for these tasks are distributed among various modules of the system (see Fig. 1). The *Dialogue Manager* generates, directs and recognizes conversations. The *Retrieval Engine* is a case-based system that uses the query that has been generated and updated by the Dialogue Manager to retrieve items from the database. The *User Modeling System* generates the initial (default) query from the user model and updates the user model based on the conversation history. The *Speech*

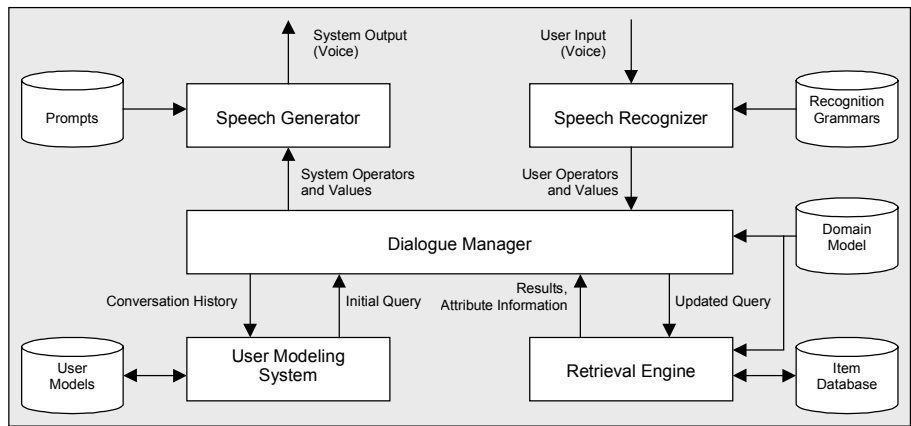


Fig.1: Overall System Architecture of the Adaptive Place Advisor

*Recognizer* and the *Speech Generator* comprise the natural language processing part of the system. We used tools from Nuance<sup>1</sup> to handle recognition and to generate appropriate prompts from a pre-recorded set.

## 5 Talking with the Driver

We view the conversational process in terms of heuristic search, similar to constraint satisfaction in that it requires the successive addition of constraints on solutions, but also analogous to game playing in that the user and system take turns. Our approach to destination advice draws heavily on an earlier analysis of the task by Elio and Haddadi (1998, 1999), which itself borrows ideas from linguistic research on speech acts (e.g., Searle, (1969)). We extend upon and adapt that work as needed to conform to the requirements of speech recognition technology and the design of the user adaptive component.

Our view of conversational recommendation as heuristic search requires us to specify the search states, operators, and operation-selection heuristics. The initial state of the search is that of a query based on the user model, where the system and user have not yet agreed upon any final attribute values. Future states, arrived at by the operators discussed below, are (more) constrained queries. A state can also consist of an over constrained query with no matching items, and the final state is reached when only a few items match the query. The search state also includes dialogue history information to help maintain a natural and coherent conversational flow.

The majority of dialogue operators are determined by the task-level goal of finding a small set of items that satisfy the user. The remaining, dialogue-level, moves are required for interactions that support progress on that task. While one side of the conversation is determined by the user, the system side of the conversation is governed by a set of control rules, described in detail in Langley, Thompson, Elio and Haddadi (1999). These rules select the next operator based on the search state. The particular instantiation of that operator (for example, which attribute to ask a question about next) is selected by consulting the Retrieval Engine and conversation history.

We group conversational actions into one operator if they achieve the same effect, so that two superficially different utterances constitute examples of the same operator if they take the dialogue in the same direction. Table 1 summarizes the operators supported by the Adaptive Place Advisor.

Let us first consider the operators available to the dialogue manager for advancing the conversation. The most obvious, ASK-CONSTRAIN, involves asking the user to provide a value for an attribute that does not yet have one. In our example conversation, we saw four examples of this operator, with the advisor asking questions about the cuisine, quality of the food, payment options, and the location (city).

In some cases, the process of introducing a constraint can produce a situation in which no items are satisfactory. When this occurs, the Dialogue Manager applies ASK-RELAX, which asks whether the user wants to drop a particular constraint.

---

<sup>1</sup> Nuance Communications, Menlo Park, CA. [www.nuance.com](http://www.nuance.com)

Another operator, SUGGEST-VALUES, answers a user's query about possible values for an attribute. In our example, this occurred in response to the inquirer's query about cuisine. Note that, in this case, the advisor lists only a few options rather than all possible choices. A similar operator, SUGGEST-ATTRIBUTES, responds to a user query about the possible characteristics of destinations.

Once the conversation has reduced the number of alternatives to a manageable size, the dialogue manager invokes RECOMMEND-ITEM, an operator that proposes a complete item to the user. Finally, the CLARIFY operator is invoked when the system is uncertain about what the user has said, either because of low speech recognition certainty, or when a value could be applicable to more than one attribute.

Now let us turn to the operators that the system assumes are available to the user. The most central action the user can take, PROVIDE-CONSTRAIN, involves specifying the value of some attribute. This can be a value for the attribute just asked for by the system, a value for a different attribute, or a replacement for a previously specified value. Our example included four instances of this operator, two in response to questions about cuisine and city, one answering a question different from the one posed by the system, and one replacing the previously provided value for cuisine. Each such answer constrains the items the system considers for presentation to the user, and thus advances the dialogue toward its goal of identifying a few satisfactory items.

As we saw above, the Place Advisor does not assume the user will always answer its questions. If the person decides that the proposed attribute is inappropriate or less

Table 1.Dialogue operators supported in the Adaptive Place Advisor

| System Operators    |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| ASK-CONSTRAIN       | Asks a question to obtain a value for an attribute                                        |
| ASK-RELAX           | Asks a question to remove a value of an attribute                                         |
| SUGGEST- VALUES     | Suggests a small set of possible values for an attribute                                  |
| SUGGEST- ATTRIBUTES | Suggests a small set of unconstrained attributes                                          |
| RECOMMEND-ITEM      | Recommends an item that satisfies the constraints                                         |
| CLARIFY             | Asks a clarifying question if uncertain about the user's most recently performed operator |
| User Operators      |                                                                                           |
| PROVIDE-CONSTRAIN   | Provides a value for an attribute                                                         |
| REJECT-CONSTRAIN    | Rejects the proposed attribute                                                            |
| ACCEPT-RELAX        | Accepts the removal of a value of an attribute                                            |
| REJECT-RELAX        | Rejects the removal of a value of an attribute                                            |
| ACCEPT-ITEM         | Accepts proposed item                                                                     |
| REJECT-ITEM         | Rejects proposed item                                                                     |
| QUERY-ATTRIBUTES    | Asks system for information about possible attributes                                     |
| QUERY-VALUES        | Asks system for information about possible values of an attribute                         |
| START-OVER          | Asks the system to re-initialize the search                                               |
| QUIT                | Asks the system to abort the search                                                       |

relevant than some other factor, he can reject the attribute or even replace it with another. The REJECT-CONSTRAIN operator captures explicit rejection. We saw this in our example when the inquirer did not specify a restaurant quality, but instead replied 'I don't care, as long as it's cheap.'

In addition, the user can explicitly accept or reject other proposals that the system makes, say for relaxing a certain attribute (ACCEPT-RELAX or REJECT-RELAX), or for a complete item (ACCEPT-ITEM or REJECT-ITEM). The user can also query about the available attributes (QUERY-ATTRIBUTES) or about possible values of that attribute (QUERY-VALUES), as we saw for cuisine. Finally, the user can reinitialize (START-OVER) or end (QUIT) the search.

## 6 Acquiring, Modeling, and Utilizing User Preferences

The conversation with the user, similar to constraint satisfaction, will ultimately direct the system to a suitable solution. However, such a conversation can become very tiring and the quality of the returned result may not be acceptable for each user.

Just as interactions with a friend who knows your concerns can be more directed and produce better results than those with a stranger, dialogues with the Adaptive Place Advisor become more efficient and effective over time. Our goal for user modeling differs from the one commonly assumed in recommendation systems, which emphasizes improving accuracy or related measures like precision and recall. We want to improve the subjective quality of both the results and the dialogue process.

While some adaptive recommendation systems (e.g. Pazzani et.al. (1996), Lang (1995), Linden, Hanks and Lesh (1997), Smyth and Cotter (1999)) require the user to provide direct feedback to generate the user model, our basic approach is to derive the preferences of the users from their interactions with the system.

To efficiently provide the users with the solution that matches their needs best, it is necessary to acquire and model the preferences of the users. A user may have preferences about:

- specific items,
- the relative importance of an attribute,
- values for an attribute,
- the combination of certain attribute-value pairs, and
- the diversity of the suggested items and values.

Item preferences manifest themselves in the user having a bias for or against a certain item, independent of its characteristics (*item preferences*). The preferences regarding an attribute represent the relative importance a user places on the attribute while selecting an item (i.e. how important is cuisine vs. price: *attribute preferences*). Preferred values show the user's bias towards certain types of items (e.g. Italian restaurants vs. French restaurants: *value preferences*) whereas preferences for certain property combinations represent certain constraints with respect to the combined occurrence of characteristics in an item (accepts Mexican restaurants only if they are cheap: *combination preferences*). While the item preferences are related to single items, the attribute, value, and combination preferences are applicable to the retrieval process in general.



If an item or a value has already been suggested in a recent interaction, it should only be suggested again after a certain time has passed. While the item, attribute, value, and combination preferences relate to the suitability of items in general, the *diversification preferences* model the suitability of an item or value at a given time.

*Item preferences* are derived by observing how often a certain item was suggested and afterwards accepted or rejected by the user. *Attribute preferences* are updated according to the item the user selects among the ones the system suggests. If the selected item was not predicted to be the most similar one to the user’s query, then the attribute preferences (i.e. weighting factors) have to be adjusted (c.f. Zhang and Yang, (1998), Bonzano, Cunningham and Smyth (1997), Wettschereck and Aha (1995), Fiechter and Rogers (2000)). *Value preferences* are calculated based on the frequencies of the values the user selects for an attribute. *Combination preferences* are

**Table 2.** Elements of a user model (without the diversification preferences)

| User Name     |                | Homer                    |        |         |         |         |         |
|---------------|----------------|--------------------------|--------|---------|---------|---------|---------|
| Attributes    | w <sub>i</sub> | Values and probabilities |        |         |         |         |         |
| Cuisine       | 0.4            | Italian                  | French | Turkish | Chinese | German  | English |
|               |                | 0.3                      | 0.2    | 0.3     | 0.1     | 0.1     | 0.0     |
| Price Range   | 0.2            | 5                        | 4      | 3       | 2       | 1       |         |
|               |                | 0.2                      | 0.3    | 0.3     | 0.1     | 0.1     |         |
| ...           | ..             |                          |        |         |         |         |         |
|               |                |                          |        |         |         |         |         |
| Parking       | 0.1            | Valet                    |        | Street  |         | Lot     |         |
|               |                | 0.5                      |        | 0.4     |         | 0.1     |         |
| Item #        |                | #0815                    | #5372  | #7638   | #6399   | ....    | .....   |
| Accept/Reject |                | 23 / 3                   | 3 / 7  | 9 / 12  | 44 / 3  | .. / .. | .. / .. |

derived by looking at the history of selected items and learning association rules. *Diversification preferences* are calculated for items and values by determining the mean time after which a value (*value diversification preferences*) or item (*item diversification preferences*) is explicitly re-selected or rejected.

Instead of modeling a diversification preference on a value level, one could envision acquiring an attribute level diversification preference. However, we think that the preference for diversity may change on a value to value basis (e.g. a person might be willing to eat Italian food much more often than Thai food). Since the value diversification preferences implicitly override the ones for attributes, we refrain from modeling diversification for attributes (e.g. one may not care about how often the price range of the suggested restaurants varies in general, but certainly about the frequency with which *expensive* restaurants are suggested).

Since the value preferences can be viewed as a probability distribution over the values for each attribute, the user model without the diversification preferences (table 2) can be used to create an initial query. In the course of the conversation, this initial query is refined and constrained with the values the user specifies for each attribute. Table 3 shows the effects of relevant dialogue operators on the query and the user model. Please note that only the user operators can update the query and the user model.

The specification or rejection of values only effects the query and not the user model directly. The user model is updated by using the last version of the query. The update is performed after the user has selected or rejected an item or in a situation in which the system is unable to find an item meeting the specifications of the user, under the assumption that the user would have accepted the proposed item, had it existed.

**Table 3.** The effects of dialogue operators on the query and the user model

| <i>Dialogue Operator</i> | <i>Effect on Query / User Model</i>                                                                                                                                                                                                                                                                                          |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACCEPT-ITEM              | <ul style="list-style-type: none"> <li>• Update value preferences based on query</li> <li>• Update attribute preferences using selected item</li> <li>• Update item preference</li> <li>• Update item and value diversification preferences</li> </ul>                                                                       |
| REJECT-ITEM              | <ul style="list-style-type: none"> <li>• Update item preference</li> <li>• Update item diversification preference</li> </ul>                                                                                                                                                                                                 |
| PROVIDE-CONSTRAIN        | <ul style="list-style-type: none"> <li>• Set probability of value for the constrained attribute in query to one</li> <li>• Set probability of other values for the attribute in query to zero</li> </ul>                                                                                                                     |
| REJECT-CONSTRAIN         | <ul style="list-style-type: none"> <li>• Drop attribute, i.e. set attribute preference (weighting factor) in query to zero</li> </ul>                                                                                                                                                                                        |
| ACCEPT-RELAX             | <ul style="list-style-type: none"> <li>• Update value preferences based on latest query</li> <li>• Update value diversification preferences based on latest query</li> <li>• Reset value preferences for the attribute in query from user model. (Dialogue Manager ensures that the question is not asked again.)</li> </ul> |
| REJECT-RELAX             | <ul style="list-style-type: none"> <li>• No effect, Dialogue Manager selects next attribute</li> </ul>                                                                                                                                                                                                                       |
| START-OVER               | <ul style="list-style-type: none"> <li>• Initialize query with user model</li> </ul>                                                                                                                                                                                                                                         |

## 7 The Retrieval Engine

The Retrieval Engine of the Adaptive Place Advisor retrieves the items that are most suitable to the users' request and match his preferences. Retrieval engines in Case-Based Reasoning systems are usually indifferent to the users' preferences. They calculate the similarity of the items in the case-base to the query using the similarity

metrics and weighting factors in the domain model. The Adaptive Place Advisor has to take the current status of the conversation and preferences of the user into account.

The current status of the conversation determines which of the attributes of the query have values associated with them. Since these values represent the user's explicit choices, we use them to generate an SQL-query to retrieve all items that match these values. The set of items that is returned from the database is used as a case-base for similarity based retrieval. This allows the content of the case-base to change between each step in the conversation.

Without the diversification preferences, the similarity between a case  $C$  and the current query  $Q$  is calculated as follows:

$$Sim(Q, C) = R_C \times \frac{\sum_{i=m}^n w_i \times P(V_{A_i})}{n - m} \quad \left| \quad (Eq. 1) \right.$$

where  $R_C$  is the user's preference for the specific case,  $w_i$  is the weighting factor (attribute preference) for attribute  $A_i$ ,  $A_m$  is the first attribute for which the user has not selected a value yet<sup>2</sup>,  $V_{A_i}$  is the value of  $A_i$  in the case, and  $P(V_{A_i})$  is the user's value preference (probability) for this value. The local similarity metric (which calculates the similarity for each attribute of the case and the query) is replaced by the probability of the user requesting the value in the case.

To take diversification preferences into account,  $R_C$  and  $P(V_{A_i})$  in equation 1 have to be extended to incorporate time effects. We define  $R_D$  and  $P_D(V_{A_i})$  as follows:

$$R_D = R_C \times \frac{1}{1 + e^{-k_R(t_C - t_R - t_{RD})}} \quad \left| \quad (Eq. 2) \right.$$

$$P_D(V_{A_i}) = P(V_{A_i}) \times \frac{1}{1 + e^{-k_V(t_C - t_V - t_{VD})}} \quad \left| \quad (Eq. 3) \right.$$

where  $t_C$  is the current time,  $t_R$  and  $t_V$  are the time when the item or value was last selected, and  $t_{RD}$  and  $t_{VD}$  are the time differences the user wants to have between having the item or value suggested again.  $R_D$  and  $P_D$  are in form of a sigmoid function where  $k_R$  and  $k_V$  determine the slope of the curve. By replacing  $R_C$  and  $P(V_{A_i})$  by  $R_D$  and  $P_D(V_{A_i})$  in equation 1, we get a similarity function which incorporates user specific diversification preferences.

The selection of the attribute to be constrained or relaxed next is based on an information gain measure. The attribute to constrain is selected by determining the attribute with the lowest entropy (highest information gain) among the attributes the user has not yet constrained. If no items were returned from the new database query, the attribute with the highest entropy (lowest information gain) with respect to the case base of the last query is selected among the attributes the user has constrained so far and suggested for relaxation. This insures that the search stays focussed and the

---

<sup>2</sup> This simplified representation assumes that the attributes  $A_1$  to  $A_{m-1}$  are the ones the user has explicitly specified during the conversation. Obviously the real system does not require the specified attributes to be in a pre-determined sequence.

smallest possible number of items is returned from the query as the new case base, i.e. that information is preserved.

Since we only need to find one item, the entropy of an attribute  $A_i$  can be calculated as:

$$H = - \sum_{V_j \in A_i} P(V_j) \times |CB_s| \times \frac{1}{|A_i = V_j|_{CB_s}} \times \log \frac{1}{|A_i = V_j|_{CB_s}} \quad \left| \quad \text{(Eq. 4)} \right.$$

where  $P(V_j)$  is the probability of the value of the attribute  $A_i$  to be  $V_j$  (this is not the probability coming from the user model, but is based on the items matching the current constraints),  $|CB_s|$  is the number of cases above a certain similarity threshold, and  $|A_i = V_j|_{CB_s}$  is the number of items in  $CB_s$  in which  $A_i$  has the value  $V_j$ .

## 8 Summary and Future Work

In this paper, we described the initial version of the Adaptive Place Advisor, an intelligent assistant designed to help people select a destination, for example a restaurant. Unlike most recommendation systems, which accept keywords and produce a ranked list, this one carries out a conversation with the user to progressively narrow his options. We also described a framework for acquiring and modeling user preferences and utilizing them for guiding the conversation and during retrieval.

Although we have a detailed design and a partial implementation of the Adaptive Place Advisor, clearly more work lies ahead. Our current similarity calculation does not take the effects of combination preferences and of diversification into account. We believe however, that these play an important role in the user's approach to selecting an item and are planning to incorporate them.

The preferences of a user may vary according to the context in which the interaction with the system is occurring. While some preferences may stay the same over various contexts, some will be overridden by specific requirements. We are planning to extend our user model to incorporate a hierarchical structure where context dependent requirements are derived from a basic user model.

Obviously we need to perform evaluations and measure the effects of the user model on the conversation and the resulting selection. We are also planning to transfer the system to similar domains (e.g. selecting books, music) and to translate the system to German.

## Acknowledgements

We thank Pat Langley, Ren   Elio, and Afsaneh Haddadi for initial conception and design of the adaptive place advisor, Cynthia Kuo and Zhao-Ping Tang for help with the implementation effort, and Stanley Peters for enlightening discussions about the design of conversational interfaces.

## References

1. Aha D., Breslow L., [Refining Conversational Case Libraries] in Leake D., Plaza E. (eds.) [Case-Based Reasoning Research and Development, Second International Conference on Case-Based Reasoning ICCBR 1997] pp. 267-278, Springer Verlag, Berlin 1997.
2. Billsus, D., Pazzani, M. (1998). [Learning collaborative information filters] Proceedings of the Fifteenth International Conference on Machine Learning (pp. 46-54). Madison, WI: Morgan Kaufmann.
3. Bonzano A., Cunningham P., Smyth B., [Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control] in Leake D., Plaza E. (eds.) [Case-Based Reasoning Research and Development, Second International Conference on Case-Based Reasoning ICCBR 1997] Springer Verlag, Berlin 1997.
4. Burke, R., Hammond, K., and Young, B. [Knowledge-based navigation of complex information spaces] In Proceedings of the 13th National Conference on Artificial Intelligence AAAI96, pp. 462-468. American Association for Artificial Intelligence, 1996.
5. Burke R., [The Wasabi Personal Shopper: A Case-Based Recommender System] in: Proceedings of the 16th National Conference on Artificial Intelligence AAAI99, American Association for Artificial Intelligence, 1999.
6. Elio R., Haddadi A., [Dialog management for an adaptive database assistant] Technical Report 98-3, Daimler-Benz research and Technology Center, Palo Alto, CA, 1998.
7. Elio R., Haddadi A., [On abstract task models and conversation policies] in Proceedings of the Agents'99 Workshop on Specifying and Implementing Conversation Policies, Seattle, WA, 1999.
8. Fiechter C. N., Rogers S., [Learning Subjective Functions with Large Margins] Proceedings of the Seventeenth International Conference on Machine Learning, June 29-July 2, 2000, Stanford University, pp. 287-294, Morgan Kaufmann Publishers, 2000
9. Gker M. H., Roth-Berghofer T., [The development and utilization of the case-based help-desk support system HOMER] Engineering Applications of Artificial Intelligence 12 (1999), pp. 665-680, Pergamon [Elsevier Science Ltd. 1999.
10. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J., [GroupLens: Applying Collaborative Filtering to Usenet News] Communications of the ACM 40,3 (1997), 77-87.
11. Lang K, [NewsWeeder: Learning to filter news] in [Proceedings of the Twelfth Conference on Machine Learning] pp.331-339, Lake Tahoe, CA, Morgan Kaufmann, 1995.
12. Langley, P. [Machine learning for adaptive user interfaces] in: [KI97: Proceedings of the 21st German Annual Conference on Artificial Intelligence] pp. 53-62. Freiburg, Germany: Springer, 1997.
13. Langley, P., Thompson, C., Elio, R. , Haddadi, A., [An adaptive conversational interface for destination advice] in: [Proceedings of the Third International Workshop on Cooperative Information Agents] pp. 347-364, Uppsala, Sweden, 1999.

14. Linden G., Hanks S., Lesh N, [Interactive Assesment of User Preference Models: The Automated Travel Assistant] in Jameson A., Paris C., Tasso C. (eds.), [User Modelling: Proceedings of the Sixth International Conference, UM97] Springer Verlag, Vienna, 1997.
15. Pazzani M., Muramatsu J., Billsus D., [Syskill and Webert: Identifying interesting web sites] in [Proceedings of the 13th National Conference on Artificial Intelligence] pp. 54-61, American Association for Artificial Intelligence, 1996.
16. Resnick P., Varian H. (eds), [Recommender Systems] Communications of the ACM, Vol. 40, No. 3, March 1997.
17. Rich, E., [User modeling via stereotypes] Cognitive Science, 3, 329-354, 1979.
18. Searle J., [Speech Acts] New York, Cambridge University Press, 1969.
19. Smyth B., Cotter P., [Surfing the Digital Wave, Generating Personalised TV Listings using Collaborative, Case-Based Recommendation] in: Althoff K. D., Bergmann R., Branting K. (eds), [Case-Based Reasoning Research and Development, Proceedings of the Third International Conference on Case-Based Reasoning ICCBR99] pp. 561-571, Springer Verlag, Berlin 1999.
20. Wettschereck D., Aha D., [Weighting Features] in M. Veloso & A. Aamodt eds., "Advances in Case-Based Reasoning, Proceedings of the First International Conference on Case-Based Reasoning ICCBR95", pp347-358, Springer Verlag, Berlin, 1995
21. Zhang Z., Yang Q., [Towards Lifetime Maintenance of Case Base Indexes for Continual Case Based Reasoning] in [Proceedings of the 1998 International Conference on AI Methodologies, Systems and Applications (AIMSA98)] Bulgaria, October 1998.

# Learning User Preferences in Case-Based Software Reuse

Paulo Gomes and Carlos Bento

Centro de Informática e Sistemas da Universidade de Coimbra  
Polo II – Pinhal de Marrocos, 3030 Coimbra, Portugal  
{pgomes, bento}@dei.uc.pt

**Abstract.** Case-Based Reasoning is a good framework for Software Reuse because it provides a flexible and powerful searching mechanism for software components. In a CBR system for software reuse it is important to learn the user preferences adapting the system software choices to the user. In a complex domain as software design, the similarity metric will also be complex, thus creating the necessity for a learning algorithm capable of weight learning. In this paper we present an evolutionary approach to similarity weight learning in a CBR system for software reuse. This approach is justified by the similarity metric complexity and recursive nature, which makes other learning methods to fail. We present experimental work showing the feasibility of this approach and we also present a parametric study, exploring several cross-over and mutation strategies.

## 1 Introduction

In a Software Reuse (Prieto-Diaz 1993; Katalagarianos and Vassiliou 1995) system the user has an important role to play. Besides using the system, the user can provide feedback to the system. This feedback can be a performance rating of the system decisions, or new knowledge about the domain. One important kind of feedback for a software reuse system is the knowledge of how to select software components or parts.

We have developed a Software Reuse system called CREATOR II, which is based on Case-Based Reasoning (CBR) (Watson 1997). In our system cases are software programs in VHDL, a hardware description language used to describe and design digital circuits. The similarity metric used in CREATOR II is crucial for retrieving the ‘right’ software components, thus being crucial for the user. This metric makes the mapping between VHDL functions and the functional requirements of the target problem, and then takes into account several issues, like code reusability, code behavior, code function, and function parameters. Our metric has also a recursive behavior allowing the recursive mapping of function structures. One of the main problems in our system is establishing the weights, which are associated to each similarity function item. It is important for the user to use the program, but for this to happen the

system must be accurate according to the user's opinion. Besides this, there is also another important issue, which is the programming style of the programmer or designer. This leads us to the need of incorporating a learning mechanism in our system, which would give feedback to the similarity metric.

After studying several learning algorithms we have chosen an Evolutionary approach (Back and Schwefel 1993) for the learning of the user preferences. In our approach, when the system provides a ranked set of software components the user can always alter the presented ranking so that it reflects the user preferences. This provides feedback to the system about the performance of the similarity metric. This feedback is then used for changing the similarity metric weights, which is done using a genetic algorithm. The main reason for using this evolutionary approach is the recursive nature and complexity of the similarity metric, which would be the reason of failure in other learning algorithms.

In the remaining of this paper we present the representation of cases in CREATOR II. We then focus in the case similarity metric, describing it, and then presenting the experimental results obtained using a case library of Small Scale Integration Circuits. Then we present other works that also learn case similarity knowledge, and finally we finish the paper with some conclusions.

## 2 Case Representation

In our system we use a Function-Behavior Case Representation (FBCR, see (Gomes and Bento 1999)) for the representation of software programs, which is based on Goel's Structure-Behavior-Function representation (Goel 1992). The FBCR formalism is intended for procedural software languages, like C, or more specific languages like VHDL, and describes a software program along two abstract levels. One is the functional level of the software program, which specifies the purposes of the design. Another is the behavior level, which describes how the functions are achieved by the structure. The structure is the software code. We now describe how function and behavior are represented in the FBCR formalism.

### 2.1 Function

Within the FBCR formalism, the functional description of a design is represented by a tree of functions. Each node of the tree represents a function. Functions are related to each other by partonomic links. This allows a function to be decomposed into sub-functions, providing a functional decomposition view of the software design.

The function tree represents two types of functions depending on being leaves of the functional tree, or at the other levels of the tree. Functions in a higher level are defined by the set of sub-functions in the descendant nodes and possibly by a behavior graph. Functions at the leaves of the tree are defined by a behavior graph. As software design problems are described at different levels of abstraction down to the instruction level, the functional description of the FBCR supports this type of reasoning.

A function is described by an identifier name, input data, output data, behavior, class, auxiliary data, and sub-functions. Input, output and auxiliary data are sets of



data objects. Data objects represent memory locations, and are in general variables, constants and parameters in the program. A data object is defined by an identifier, a data class (for example the variable data type) and a set of properties. A property is described by the property name, value and measure units. Input and output data objects represent, respectively, the function input and output parameters. Usually, associated with an output data parameter there is an output expression, which corresponds to a possible final expression value of the parameter. Not all output parameters have output expressions, depending mainly in the abstraction level of the function to which the parameter belongs. Associated to each function there is a class used to categorize functions and to help the designer specifying the problem.

## 2.2 Behavior

The behavior of a function is described by a graph, where a node represents a behavior state, and an edge represents a transition between behavior states. A behavior graph represents the program execution flow, from the initial behavior state to the final behavior states, along which data objects are transformed. Data object properties can change, or data objects can be created or eliminated. A behavior graph is identified by an identifier and its initial state.

A behavior state represents the system state at a specific instant of time and is defined by an identifier, data objects, and the transitions that have this state as the source state. Behavior transitions represent the causes and constraints for state changing. An identifier, a source state, a destination state, a set of causes, and a set of constraints define each behavior transition.

Behavior transitions have two types of attributes: constraints and causes. In the first case, constraints represent boolean expressions that must evaluate to true in order for the transition to occur. There are two types of constraints in this category: data and property constraints. A data constraint is a Boolean expression defined by a data object, a relational operator and a value. This type of constraint states that the data object value must comply with the constraint defined by the relational operator and value in order for the transition to occur. The relational operator and the value are optional; if they are absent, the constraint means that the data object must exist. Property constraints are defined by a data object, a property, a relational operator and a value. This type of constraint states that the data object must comply with a specific property value. In the case that the relational operator and value are omitted, the constraint implies the existence of the data object property.

Causes of transitions are named primitive functions. These are low-level functions representing language instructions, operators or pre-defined functions. The primitive functions play an important role in the FBCR formalism, connecting the behavior level to the structural level.

## 3 Similarity Metric

In CREATOR II the user specifies a set of functional specifications (new problem) that are used as probes to search the case library of previous software designs. Then

the system retrieves a set of cases ordered by the similarity with the new problem. In order to rank the retrieved cases, CREATOR II uses a similarity function. As part of this function there is a reusability measure that estimates the amount of effort required for case reuse. The reusability metric is part of function similarity, which in turn is part of case similarity.

### 3.1 Case Similarity Metric

The case similarity metric has three aspects: structural similarity, sub-function similarity, and functional similarity. To access the similarity of two cases a mapping between the two cases must be done. The mapping is performed by levels, starting in the higher level functions of the problem description, recursively going down in the functional trees. The mapping establishes for each function in the problem description (in the same tree level) a matching function in the retrieved case. There can be unmatched functions in the problem or in the retrieved case. The mapping algorithm is based on the distance between the functions classes. The algorithm will map the functions which have the most similar function classes.

The algorithm for computing the case similarity metric is:

```

CASE_SIM (case_functions, problem_functions)
  MAP_FUNCTIONS(case_functions,problem_functions)
  case_sim  $\leftarrow$  0
  For Each Mapped Pair of Functions (F1, F2)
    case_sim  $\leftarrow$  case_sim +  $\alpha$  * SIM(F1,F2)
    sub_functions1  $\leftarrow$  GET_SUBFUNCTIONS(F1)
    sub_functions2  $\leftarrow$  GET_SUBFUNCTIONS(F2)
    case_sim  $\leftarrow$  case_sim +  $\delta$  * CASE_SIM (sub_functions1,sub_functions2)
  End For
  case_sim  $\leftarrow$  case_sim +  $\beta$  * (#unmatched_functions)/(MAX(#case_functions,#problem_functions))

```

In the algorithm *case\_functions* and *problem\_functions* are respectively the retrieved case functions and the new problem functions. The first time that this algorithm is called these two variables must have the high level functions of the case and the problem. The first step of the algorithm is to map the functions, after that the *case\_sim* variable is initialized to zero. This variable is going to store the case similarity result. Then a cycle is used for computing the function similarity (function SIM, that will be described in the next subsection) between each mapped function, and its sub-function similarity (the recursive call of CASE\_SIM, with the sub-functions of each mapped function as parameters). This recursive call enables the computation of the structural similarity between cases. The final step in computing the case similarity is using the number of unmatched functions (see last line in the algorithm).  $\alpha$ ,  $\delta$ , and  $\beta$  are weights associated respectively with the function similarity, the subfunction similarity, and the function structure similarity.

### 3.2 Function Similarity Metric

The function similarity is computed based on the class distance, parameter similarity, and code complexity (or reusability metric). Functions in the same class are supposed

to have the same functionality. The parameter similarity is computed based on the differences between the function parameters. Code complexity is a measure of function reusability (in the next subsection we will describe this metric in detail). The function similarity formula between  $F1$  (case function) and  $F2$  (problem function) is the following:

$$\text{Sim}(F1, F2) = W1 * [1 - \text{ClassDistance}(C1, C2)] + W2 * \text{SimParams}(F1, F2) + W3 * \text{Reusability}(F1) \quad (1)$$

$W1$ ,  $W2$  and  $W3$  are weights associated with the function class similarity, parameter similarity, and function reusability.  $C1$  and  $C2$  are the function classes of  $F1$  and  $F2$ .

The function class similarity ( $\text{ClassDistance}$ ) is the minimum path distance between the classes. A path distance between two classes has a length equal to the sum of the arc weights between these two classes. Weights for an arc are computed using a search algorithm that when it has to ascend in the class taxonomy it gives a weight  $w1$  to the arc, and when descending it gives the arc a  $w2$  weight. What is computed by this distance is the number of specializations and generalizations needed to reach one function class, starting at another class. The formula for computing the similarity between function classes  $C1$  and  $C2$  is,

$$\text{ClassDistance}(C1, C2) = 1 - [1 / (\log(\text{MinPath}(F1, F2) + 1) + 1)] \quad (2)$$

Parameter similarity is based on the classification of the parameters within three different classes: input control, input data, and output data. An input control parameter is a parameter used in the constraints of a behavior transition of the function behavior, in other words, is part of a condition that influences the program control flow. A parameter can also be input control if it is used to compute the value of a variable present in a constraint of a behavior transition.

A parameter is considered input data if it is used to compute the value of an output parameter. This is easily detected in the behavior graph of the function whose parameters are being classified. But a parameter can also be input data if it contributes indirectly to the value of an output parameter, that is, if it is used to compute the value of the output parameter through the use of other variables.

The last class of parameters is output data, which comprises the output parameters whose values are changed within the function. A parameter can also be input and output at the same time. The way that a parameter is classified as input or output is inferred from the program code.

The classification of function parameters is important for the similarity metric, because it introduces the behavior aspect into account. When the user classifies the parameters associated with the desired functions, s/he is also stating what role is the parameter going to perform in the function.

To compute parameter similarity between two functions,  $F1$  and  $F2$ , we use the following function:

$$\text{SimParams}(F1, F2) = Z1 * \text{SimPar}(F1\_IC, F2\_IC) + Z2 * \text{SimPar}(F1\_ID, F2\_ID) + Z3 * \text{SimPar}(F1\_OD, F2\_OD) \quad (3)$$

$F1\_IC, F2\_IC$  are the set of input control parameters in  $F1$  and  $F2$  respectively,  $F1\_ID, F2\_ID$  input data, and  $F1\_OD, F2\_OD$  output data.  $Z1, Z2$  and  $Z3$  are weights associated with each class of parameters.

SimPar is the function similarity between two sets of parameters of the same type, and is given by the following formula:

$$\text{SimPar}(X,Y) = T1 * (\text{MappingPar}(X,Y) / \text{Mapped}) + T2 * (\text{UnMapped} / (\text{Mapped} + \text{UnMapped})) \quad (4)$$

This function is computed using the mapping between parameters.  $T1$  and  $T2$  are weights associated with the mapped parameters and unmapped parameters respectively.  $\text{Mapped}$  and  $\text{UnMapped}$  are respectively, the number of mapped parameters and unmapped parameters.

The mapping is performed by the MappingPar function. This function tries to map a parameter in one set to a parameter of the other set. This mapping is done using two characteristics of parameters: the data class to which the parameter belongs, and the data length of this parameter. The data length refers to data objects which are arrays or vectors and have a data length associated with it. According to these two attributes the mapping function tries to match the parameters with the same data class and data length, then it tries to match the remaining parameters with the same data class, and finally it tries to match the remaining parameters using the distance between two data classes. A parameter can only be mapped once, so if the sets have different number of parameters, there will be unmapped parameters. The mapping function is given by:

$$\text{MappingPar}(X,Y) = \text{Sum of all Map}(x,y) \quad (5)$$

The  $\text{Map}(x,y)$  is the mapping function between parameter  $x$  and parameter  $y$ . The mapping is based on the parameter class similarity. Parameters with the same classes are mapped. This function is given by:

$$\text{Map}(x,y) = CS1 * (1 - \text{ClassDistance}(x,y)) + CS2 * ((1 - |Lx - Ly|) / \text{Max}(Lx, Ly)) \quad (6)$$

$CS1$  and  $CS2$  are weights concerning the distance between data classes, and the difference between data lengths.  $\text{ClassDistance}(x,y)$  is the distance in the data class taxonomy between data class of  $x$  and data class of  $y$ .  $Lx$  and  $Ly$  are the data length of  $x$  and  $y$  respectively.

If the type of parameters being mapped is output data, then the comparison between the output expressions of both parameters is made. This is done adding a new term to formula (6), which is:

$$CS3 * [1 - \text{OutputExpDistance}(x,y)]$$

$\text{OutputExpDistance}(x,y)$  is the distance between the output expressions of both parameters, and  $CS3$  is the weight associated with the output expression similarities.

### 3.3 Function Reusability Measure

The reusability metric is used in the function similarity metric to estimate the function reusability. This measure relies in three different function characteristics: the number of code lines of the function implementation, the number of comment lines in the

function file, and the measure of the complexity of the function's behaviour graph. Suppose that the reusability of *FI* is to be computed, then:

$$\text{Reusability}(FI) = K0 / (K1 * \text{Code} + K2 * \text{Behavior\_Complexity} + K3 * \text{Comments}) \quad (7)$$

*Code* and *Comments* are respectively, the number of lines in the function code, and the number of comment lines in the code. We have chosen the number of lines of comments in the function file, because much of the code files that were analyzed, use global comments concerning all the functions in the file, and not just local comments to the function. Besides this, comments related to a specific function are very hard to identify inside a file, they can be anywhere. *K0*, *K1*, *K2* and *K3* are weights associated to each part of the formula.

*Behavior\_Complexity* estimates the complexity of the function. This measure is given by the sum of all the function behaviour graphs, plus the sum of each graph complexity for each sub function. This is used in the function similarity metric. The complexity of a behaviour graph is given by the following formula:

$$\text{Graph\_Complexity} = C1 * \text{Nodes} + C2 * \text{Edges} + C3 * \text{Ramifications} + C4 * \text{Cycle\_Nodes} \quad (8)$$

*Nodes* is the number of behavior states in the graph. *Edges* is the number of behavior transitions in the graph. *Ramifications* is the number of behavior states which have more than one transition starting from it. *Cycle\_Nodes* is the number of behavior states that make part of a cycle. *C1*, *C2*, *C3* and *C4* are weights associated with each attribute.

## 4 Genetic Algorithm

As seen before the similarity function is complex and recursive, making it hard to use normal optimization techniques for weight learning. Due to these difficulties Genetic Algorithms (GAs) are a good candidate algorithm for performing the weight learning. GAs have global search capabilities, and are domain independent, enabling them to work with the recursive and domain dependent nature of our similarity function. Having the user rankings as training feedback, we are able to use a genetic algorithm for finding a new weight configuration reflecting the user preferences.

The genetic algorithm used is:

```

Generate the initial population P0
Evaluate P0
While not stopping criteria Do
    Select ( Pt → Pt+1 )
    Recombine ( Pt+1 → P't+1 )
    Mutate ( P't+1 → P''t+1 )
    Evaluate ( P''t+1 )
End While

```

In the remaining of this section we will present the individual representation, the genetic operators, and the evaluation criteria.

## 4.1 Individual Representation

In our approach a population is a set of individuals, where each individual represents a weight configuration. Each individual has six different genes. Each gene represents a weight in the format of a real number between zero and one. The weights used are: the function similarity ( $\alpha$ ), the function structure similarity ( $\beta$ ), the subfunction similarity ( $\delta$ ), the function class similarity (W1), the function parameter similarity (W2), and the function reusability (W3). These parameters were chosen because they are the most important ones in the case similarity metrics. The other weights are set by the user.

There are more than one stopping criteria in our algorithm. The user can select from the following criteria: individual satisfies a threshold error regarding the user ranking, the algorithm has reached the maximum number of iterations, or the algorithm has reached the maximum time limit.

## 4.2 Genetic Operators

We use three genetic operators: selection, recombination, and mutation. The selection operator chooses the next generation individuals and is implemented as a *q-tournament* selection strategy. If the population size is  $\mu$  and the current population is  $P_t$ , then the algorithm is going to select  $\mu$  times the best individual from a set of individuals. This set is a randomly chosen set from  $P_t$ , with  $q$  elements. The parameter  $q$  is a user defined parameter. The selected individuals make the new population  $P_{t+1}$ .

The recombination operator recombines two individuals chosen from the  $P_{t+1}$  population yielding two new individuals replacing their parents in  $P_{t+1}$ . The individuals to be recombined are chosen using a *q-tournament* selection. There are three types of recombination implemented in CREATOR II: discrete, uniform, and non-uniform. The discrete recombination tweaks genes according to the recombination probability for a gene. Genes can only be exchanged if they correspond to the same position within the individual, i.e., the same gene. This is done for all genes of the selected individuals. The uniform recombination strategy for each gene of two individuals A and B, computes the new genes ( $w'_{a,i}$  and  $w'_{b,i}$ ) by the following expression:  $w'_{a,i} = \kappa * w_{b,i} + (1-\kappa) * w_{a,i}$ . Where  $\kappa$  is a parameter established by the user. The non-uniform strategy is the same as the uniform one, except that the  $\kappa$  parameter is not a fixed one, but it varies according to the function  $\Delta(t,1)$ , where  $t$  is the population number. The function  $\Delta$  is equal to:  $1-r^s$ , where  $s$  is equal to  $(1-(t/T))^b$ .  $r$  is a random number between zero and one.  $T$  is the maximal generation number.  $b$  is a system parameter determining the degree of dependency on the iteration number.

The mutation operator changes a gene randomly according to the mutation probability for a gene. This enables small changes in the weights performing a search in the near-by search space. We use four different mutation operators: random replacement, uniform, non-uniform, and geometric. The random replacement mutation replaces the gene value with a random one. The uniform mutation changes a gene  $w'_{a,i}$  in the following way: Generate random number 0 or 1; then If random number is 0 Then  $w'_{a,i} = w_{a,i} + \lambda$  Else  $w'_{a,i} = w_{a,i} - \lambda$ , where  $\lambda$  is a user defined number, and  $w_{a,i}$

is the current gene value. The non-uniform mutation is similar to the uniform mutation, except that the  $\lambda$  constant is replaced by the value of  $\Delta(t, UB - wa_i)$  if it is going to be added, or  $\Delta(t, wa_i - LB)$  if it is going to be subtracted. UB and LB are respectively the upper bound and lower bound values of  $wa_i$ . The function  $\Delta$  is the same as defined above for the mutation. The geometric mutation is simply defined by  $w'a_i = wa_i * \Delta(t, 1)$ .

### 4.3 Evaluation Function

The evaluation function has the goal of accessing each weight configuration (individual) performance in relation to the user preferences. In this process the user feedback is crucial, and is gathered when the user reorders the case rankings presented by the system. CREATOR II stores all the case rankings that the user altered, along to the problem associated to the respective ranking. To evaluate an individual the following algorithm is used,

```

For each ranking stored as user feedback Do
  For each case in the ranking Do
    Compute the case ranking using the individual weight configuration
  End For
  Rank the cases accordingly to the similarity score
  Compute the ranking error based on the computed ranked and the user ranking
End For
Compute the average error

```

The ranking error is computed based on the difference between the ranking computed using the individual weights and the user ranking. The error is equal to the sum of errors for each case ranking. Computing the ranking error for case  $C_n$ , is given by:  $(N - S_{user}(C_n)) * |S_{user}(C_n) - S_{system}(C_n)|$ . Where  $S_{user}(C_n)$  is the position of  $C_n$  in the user ranking,  $S_{system}(C_n)$  is the position of  $C_n$  in the system ranking, and  $N$  is the number of cases in the user ranking. The  $(N - S_{user}(C_n))$  factor assures that high-ranked functions in the user ranking have more importance than low-ranked functions.

There are several issues related with this evaluation function. A first one is the dynamic aspect of the case library. If a case used in a ranking is deleted from the case library, the associated error is going to be greater, not necessarily corresponding to a worse ranking by the system. On the other hand, cases added to the case library can also alter the error function for a ranking. Other issue has to do with the performance of the algorithm. The evaluation function is very complex, which degrades time performance. In CREATOR II the time performance is crucial, which leaves us with a dilemma. To improve performance we added a cache for function similarity computations, nevertheless the evaluation algorithm still has a insufficient performance. This makes the selection of the right parameters and strategies determinant for achieving an acceptable performance. In the next section we make a parameter study so that the right parameters can be selected. The last but determining issue is the quality of the rankings given by the user. When the rankings provided by the user have inconsistencies the algorithm does not find a zero error configuration because it is not possible with these rankings. In this situation the algorithm finds the individual which presents

the less error, but a possible alternative solution is to warn the user to review the provided rankings.

## 5 Experimental Work

The experiments consisted in a parametric study of the mutation and crossover strategies. Our objective was to determine which are the best strategies to use in our test domain. For this a case library comprising 98 VHDL cases describing SSI (Small Scale Integration Circuits) was used. This case library describes from simple logical gates, to complex ALUs (Arithmetic Logic Units), passing by multiplexers and encoders. To test the learning mechanism we devised ten different problems. Each of these problems was run in CREATOR II's retrieval mechanism, which gave a ranked set of cases. The user was then presented with the ranking giving his feedback on the presented ranking by re-ranking the cases. This feedback was then saved for further use by the learning mechanism.

After the gathering of the user's feedback the learning algorithm was used to find the best weight configuration. Then, the algorithm used the ten rankings (one for each problem) established by the user. The experiments have the goal of studying the mutation and crossover strategies. For the mutation strategy we have four different strategies: random, uniform, non-uniform, and geometric. The crossover strategy can have three types: discrete, uniform and non-uniform. For each mutation/crossover pair we performed ten different runs of the learning algorithm, gathering the best individual scores.

The parameters used for each of these runs are: population size (10), tournament selection parameter (3), dependency (5), mutation probability for a gene (0.01), fixed value for the uniform mutation (0.1), crossover probability (0.75), fixed value for the uniform crossover (0.5). Each algorithm run had the time limit of 200 seconds, which corresponds to more or less to 64 generations. There are one important issue here, which is the system performance. The evaluation is a very time consuming part of the algorithm making performance decisions crucial to the usage of the system. So, we devised a short run of the algorithm testing its performance in situations where the user would be waiting for the system to learn the new weight configuration. This decision made us select a population size of 10, and selecting 200 seconds as time limit for the learning algorithm. Despite this short time limit the algorithm managed to find most of the times an optimal solution (an individual with a ranking error of zero).

Figure 1 presents the average ranking error of the best individual (the one with lowest ranking error) found in each pair of mutation/crossover strategy. It can be clearly seen that the uniform crossover strategy produces the best results, compared with the other crossover strategies. Also from the graphic analysis we can observe that the best mutation strategy is the non-uniform strategy.



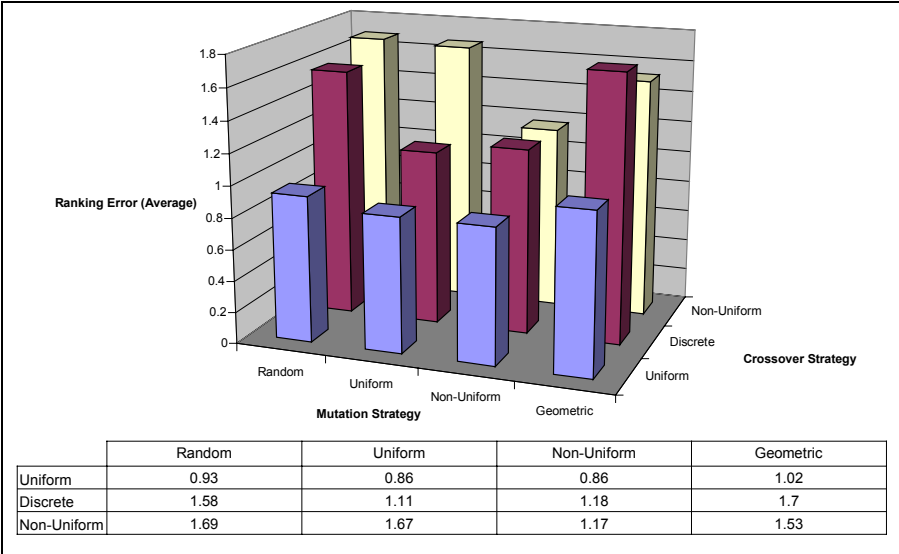


**Figure 1.** Average of the ranking error of the best individual found for each pair of mutation/crossover strategies

## 6 Related Work

There are several approaches to weight learning in similarity metrics used in case-based reasoning systems. Most of these are approaches used for learning the weights associated with each case feature. For example, Ricci and Avesani (Ricci and Avesani 1995) use a reinforcement learning algorithm to learn the local similarity metrics. Wetschereck and Aha (Wetschereck and Aha 1995) evaluated several methods that automatically assign weight settings with little or no domain-specific knowledge. They have also introduced a five-dimensional framework that categorizes automated weight-setting methods. Fox and Leake (Fox and Leake 1995) use introspective reasoning to learn how to refine indexing. The introspective reasoning component in a CBR system monitors system performance, detecting poor retrievals and refining the indexing criteria. Their system is capable of learning features implicit in the original problem description but not explicitly used in the initial indexing scheme. Another important work is the one performed by Bonzano et. al. (Bonzano, Cunningham et al. 1997), in which they present a case study in Air Traffic Control using introspective learning to improve retrieval. With this technique they are able to learn the most important features for retrieval. Bruninghaus (Bruninghaus and Ashley 1997) has also developed methods to automatically index cases described in text, using machine learning for assigning indices to textual cases. Most of these works are centered in K-Nearest Neighbor (K-NN) similarity metric. The similarity metric presented here is different from a K-NN similarity metric, specially in three fundamental aspects: the function is much more complex, it is a recursive function, and the weights are not necessarily associated with a case feature, they can be associated with a set of features.

## 7 Conclusions

In a CBR system for software reuse it is important to learn the user preferences adapting the system to the user and not the opposite. We must always have in mind that without the user a software reuse system is worthless, because its goal is to aid the programmer. Having a complex domain as software design in mind, our similarity metric is complex and recursive, thus making us explore other learning algorithms for weight learning.

In this paper we present an evolutionary approach to similarity weight learning in a CBR system for software reuse. This approach is justified by the similarity metric complexity and recursive nature, which makes other learning methods to fail. In the presented experiments we show the feasibility of this approach and we also present a parametric study, exploring several crossover and mutation strategies. From this study we conclude that the uniform crossover strategy provides the best results. From the view point of the mutation strategy, the non-uniform strategy was found to be the best of the mutation strategies.

## References

- Back, T. and H.-P. Schwefel (1993). „An Overview of Evolutionary Algorithms for Parameter Optimization.“ *Evolutionary Computation* **1**(1): 1-23.
- Bonzano, A., P. Cunningham, et al. (1997). *Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control*. International Conference on Case-Based Reasoning (ICCBR'97), Providence, RI, USA, Springer-Verlag.
- Bruninghaus, S. and K. Ashley (1997). *Using Machine Learning for Assigning Indices to Textual Cases*. Second International Conference on Case-Based Reasoning (ICCBR'97), Providence, RI, USA, Springer-Verlag.
- Fox, S. and D. Leake (1995). *Learning to Refine Indexing by Introspective Learning*. International Conference on Case-Based Reasoning (ICCBR'95), Sesimbra, Portugal, Springer-Verlag.
- Goel, A. (1992). Representation of Design Functions in Experience-Based Design. *Intelligent Computer Aided Design*. D. Brown, M. Waldron and H. Yosnikawa, Elsevier Science.
- Gomes, P. and C. Bento (1999). *Converting Programs into Cases for Software Reuse*. International Joint Conference on Artificial Intelligence (IJCAI 99) Workshop: Automating the Construction of Case-Based Reasoners, Stockholm, Sweden.
- Katalagarianos, P. and Y. Vassiliou (1995). „On the reuse of software: a case-based approach employing a repository.“ *Automated Software Engineering* **2**: 55-86.
- Prieto-Diaz, R. (1993). „Status Report: Software Reusability.“ *IEEE Software*(May).
- Ricci, F. and P. Avesani (1995). *Learning a Local Similarity Metric for Case-Based Reasoning*. International Conference on Case-Based Reasoning (ICCBR'95), Sesimbra, Portugal, Springer-Verlag.
- Watson, I. (1997). *Applying Case-Based Reasoning: techniques for enterprise systems*, Morgan Kaufmann.
- Wettschereck, D. and D. Aha (1995). *Weighting Features*. International Conference on Case-Based Reasoning (ICCBR'95), Sesimbra, Portugal, Springer-Verlag.

# A Method for Predicting Solutions in Case-Based Problem Solving

Eyke Hüllermeier

Institut de Recherche en Informatique de Toulouse  
Université Paul Sabatier  
eyke@irit.fr

**Abstract.** In order to predict the solution to a new problem we proceed from the “similar problem–similar solution” assumption underlying case-based reasoning. The concept of a *similarity hypothesis* is introduced as a formal model of this meta-heuristic. It allows for realizing a constraint-based inference scheme which derives a prediction in the form of a set of possible candidates. We propose an algorithm for learning a suitable similarity hypothesis from a sequence of observations. Basing the inference process on hypotheses thus defined yields (set-valued) predictions that cover the true solution with high probability. Our method is meant to support the overall (case-based) problem solving process by bringing a promising set of possible solutions into focus.

## 1 Introduction

The idea that reasoning and problem solving is guided by experiences from situations which are *similar* to the current one has a long tradition in philosophy and cognitive psychology. Being a major assumption of artificial intelligence, it has recently been realized at a formal level in fields such as, e.g., machine learning and case-based reasoning (CBR). Instance-based learning (IBL) algorithms [3], for example, classify a new instance based on the assumption that “similar instances have similar classifications.” Likewise, problem solving in CBR relies upon the hypothesis that “similar problems have similar solutions” [12]. This (heuristic) assumption and variations thereof will subsequently be referred to as the “CBR hypothesis.”

Apparently few attempts at formalizing the CBR hypothesis in a systematic way and, thus, at contributing to a theoretical foundation of CBR have as yet been made [5,6,14]. In this paper, we extend a formalization which has been proposed in [9]. This approach proceeds from a constraint-based interpretation of the CBR hypothesis, according to which the similarity between problems imposes a constraint on the similarity between associated solutions in the form of a lower bound. Our focus is on *prediction* as a performance task [5,7] and essentially concerns the RETRIEVE and REUSE processes within the (informal)  $R^4$  model of the CBR cycle [1]. For pointing out the restriction to a certain aspect of CBR, this method is referred to as *case-based inference* (CBI). It is closely related to lazy learning algorithms [2], particularly those which are

derivatives of the  $k$ -NEAREST NEIGHBOR ( $k$ NN) classifier [4]. Still, there are also important methodological differences, as will be seen below. Besides, CBI is not intended as a special performance task such as, e.g., classification [3] or function learning [10].

A basic concept of CBI is that of a *similarity hypothesis*, which defines a formal model of the CBR assumption. In this paper, we consider the problem of acquiring similarity hypotheses suitable for the application at hand. More precisely, we propose an algorithm which learns a hypothesis from a sequence of observations. Similarity hypotheses thus obtained allow for deriving (set-valued) predictions which are correct with high probability.

The remaining part of the paper is organized as follows: The next section reviews and extends the constraint-based approach to CBI proposed in [9]. An algorithm for learning similarity hypotheses will be proposed in Section 3. In Section 4, we shall take a closer look at formal properties of this algorithm. The paper concludes with a brief summary and some remarks.

## 2 Case-Based Inference

We define the primitive concept of a *case* (observation) as a tuple consisting of a *situation* and a *result* or *outcome* associated with the situation.<sup>1</sup>

**Definition 1 (CBI setup).** A CBI setup is defined as a 6-tuple

$$\Sigma = \langle \mathcal{S}, \mathcal{R}, \varphi, \sigma_{\mathcal{S}}, \sigma_{\mathcal{R}}, \mathcal{M} \rangle,$$

where  $\mathcal{S}$  is a countable set of situations,  $\mathcal{R}$  is a set of results, and  $\varphi : \mathcal{S} \rightarrow \mathcal{R}$  assigns results to situations. The functions  $\sigma_{\mathcal{S}} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$  and  $\sigma_{\mathcal{R}} : \mathcal{R} \times \mathcal{R} \rightarrow [0, 1]$  define (reflexive, symmetric, and normalized) similarity measures over the set of situations and the set of results, respectively.  $\mathcal{M}$  is a finite memory  $\mathcal{M} = (\langle s_1, r_1 \rangle, \langle s_2, r_2 \rangle, \dots, \langle s_n, r_n \rangle)$  of cases  $c = \langle s, \varphi(s) \rangle \in \mathcal{S} \times \mathcal{R}$ .

We denote by  $\mathcal{M}^\downarrow$  the projection of the memory  $\mathcal{M}$  to  $\mathcal{S}$ , i.e.,  $\mathcal{M}^\downarrow = (s_1, \dots, s_n)$ . Moreover, we shall treat  $\mathcal{M}$  as a sequence rather than a set of cases. Still, we retain the standard notations for operations on sets. For example,  $(s_1, \dots, s_n) \cup (s'_1, \dots, s'_m)$  defines the list  $(s_1, \dots, s_n, s'_1, \dots, s'_m)$  of (not necessarily different) situations. The sets

$$\begin{aligned} D_{\mathcal{S}} &\doteq \{ \sigma_{\mathcal{S}}(s, s') \mid s, s' \in \mathcal{S} \}, \\ D_{\mathcal{R}} &\doteq \{ \sigma_{\mathcal{R}}(\varphi(s), \varphi(s')) \mid s, s' \in \mathcal{S} \} \end{aligned}$$

denote the sets of actually attained similarity degrees.

Even though we assume that situations determine outcomes, the derivation of results might involve a computationally complex process. In this connection,

<sup>1</sup> For reasons of generality, these expressions are preferred to the commonly used terms “problem” and “solution.”

we understand *case-based inference* as a method supporting the overall process of problem solving by bringing possible results of a new situation into focus. Thus, given a CBI problem  $\langle \Sigma, s_0 \rangle$  consisting of a setup  $\Sigma$  and a new situation  $s_0 \in \mathcal{S}$ , the task is to predict the result  $r_0 = \varphi(s_0)$  associated with  $s_0$ . To this end, CBI performs according to the CBR *principle*: It exploits experience represented by precedent cases to which it “applies” background knowledge in the form of the heuristic CBR hypothesis.

## 2.1 Similarity Profiles

The structure of the system under consideration,  $(\mathcal{S}, \mathcal{R}, \varphi)$ , is completely determined by the mapping  $\varphi$ . The following definition provides a characterization of the *similarity structure* of this system.

**Definition 2 (similarity profile).** *The function  $h_\Sigma : D_\mathcal{S} \rightarrow [0, 1]$  defined by*

$$h_\Sigma(x) \doteq \inf_{s, s' \in \mathcal{S}, \sigma_\mathcal{S}(s, s')=x} \sigma_\mathcal{R}(\varphi(s), \varphi(s'))$$

*and called the similarity profile of the CBI setup  $\Sigma$ .*

The similarity profile  $h_\Sigma$  is the “fingerprint” of the system  $(\mathcal{S}, \mathcal{R}, \varphi)$  at the *similarity level*. Just like  $\varphi$  determines dependencies at the instance level,  $h_\Sigma$  depicts dependencies between degrees of similarity: Given the similarity of two situations, it provides a lower bound to the similarity of the respective outcomes. It hence conveys a precise idea of the extent to which the application at hand actually meets the CBR hypothesis. In fact, the stronger the similarity structure of  $(\mathcal{S}, \mathcal{R}, \varphi)$  is developed, the more constraining the similarity profile will be.

The concept of a similarity profile is intended to support a sound inference process, as will be seen later. However, the specification of lower similarity bounds which are *globally* valid, i.e., which hold true for all pairs of cases, might appear unnecessarily cautious under certain conditions. This motivates two generalizations of Definition 2 which we shall introduce now.

Problems in CBR might be solved repeatedly using the same memory  $\mathcal{M}$  of cases, or at least a memory which does not change frequently. We may benefit from this fact by adjusting the model of the similarity structure to  $\mathcal{M}$ . This leads to the idea of replacing a similarity profile by an  $\mathcal{M}$ -similarity profile.

**Definition 3 ( $\mathcal{M}$ -similarity profile).** *The function  $h_\Sigma^\mathcal{M} : D_\mathcal{S} \rightarrow [0, 1]$  defined by*

$$h_\Sigma^\mathcal{M}(x) \doteq \inf_{s \in \mathcal{M}^\perp, s' \in \mathcal{S}, \sigma_\mathcal{S}(s, s')=x} \sigma_\mathcal{R}(\varphi(s), \varphi(s'))$$

*is called the  $\mathcal{M}$ -similarity profile of  $\Sigma$ .*

A second generalization is motivated by the observation that the CBR assumption is often not satisfied equally well for all parts of the instance space  $\mathcal{S} \times \mathcal{R}$ .<sup>2</sup>

<sup>2</sup> In a game playing context, for instance, the CBR hypothesis hardly applies to certain “tactical” situations [15].

Again, the global validity of the similarity profile might be an obstacle to the definition of tight bounds for those regions where the CBR hypothesis does actually apply rather well. In order to avoid this problem we maintain an individual similarity profile for each case in the memory. This approach is somehow comparable to the use of *local metrics* in *k*NN algorithms and IBL, e.g., metrics which allow feature weights to vary as a function of the instance [16]. It leads us to introduce the concept of a *local similarity profile*.

**Definition 4 (local similarity profile).** Let  $s \in \mathcal{S}$ . The function  $h_\Sigma^s : D_\Sigma \rightarrow [0, 1]$  defined by

$$h_\Sigma^s(x) \doteq \inf_{s' \in \mathcal{S}, \sigma_\Sigma(s, s') = x} \sigma_\Sigma(\varphi(s), \varphi(s'))$$

is called the *local similarity profile* associated with  $s$ , or the *s-similarity profile* of  $\Sigma$ . A collection  $h_\Sigma^\mathcal{M} = \{h_\Sigma^s \mid s \in \mathcal{M}^\downarrow\}$  of local profiles is referred to as the *local  $\mathcal{M}$ -similarity profile*.

A local profile indicates the validity of the CBR hypothesis for *individual* cases. It might hence serve as a (complementary) criterion for selecting “competent” cases to be stored in the memory [18]. In fact, the next section will show that a situation with a poorly developed profile hardly contributes to precise predictions.

The following relations hold between the different types of similarity profiles:

$$h_\Sigma = \bigwedge_{s \in \mathcal{S}} h_\Sigma^s, \quad h_\Sigma^\mathcal{M} = \bigwedge_{s \in \mathcal{M}^\downarrow} h_\Sigma^s.$$

That is,  $h_\Sigma$  and  $h_\Sigma^\mathcal{M}$  are lower envelopes of the class of local profiles associated with situations in  $\mathcal{S}$  and  $\mathcal{M}^\downarrow$ , respectively. Consequently,  $h_\Sigma \leq h_\Sigma^\mathcal{M} \leq h_\Sigma^s$  for all memories  $\mathcal{M}$  and situations  $s \in \mathcal{M}^\downarrow$ .

## 2.2 Similarity Hypotheses

Needless to say, knowledge about the similarity profile of a setup  $\Sigma$  will generally be incomplete. This motivates the related concept of a *similarity hypothesis*, which is thought of as an approximation of a similarity profile. It thus defines a formal model of the CBR hypothesis, adapted to the system under consideration.

**Definition 5 (similarity hypothesis).** A *similarity hypothesis* is identified by a function  $h : [0, 1] \rightarrow [0, 1]$ . The intended meaning of the hypothesis  $h$  is that

$$\sigma_\Sigma(s, s') = x \Rightarrow \sigma_\Sigma(\varphi(s), \varphi(s')) \geq h(x) \quad (1)$$

holds true for all  $s, s' \in \mathcal{S}$ . A hypothesis  $h$  is called *stronger* than a hypothesis  $h'$  if  $h' \leq h$  and  $h \not\leq h'$ . We say that  $h$  is *admissible* for the CBI setup  $\Sigma$  if  $h(x) \leq h_\Sigma(x)$  for all  $x \in D_\Sigma$ .

Now, consider a CBI problem  $\langle \Sigma, s_0 \rangle$  and suppose that the hypothesis  $h$  is admissible. Then, the following restriction holds true:

$$\varphi(s_0) \in \widehat{\varphi}_{h, \mathcal{M}}(s_0) \doteq \bigcap_{s \in \mathcal{M}^\perp} \mathcal{N}_{h(\sigma_{\mathcal{S}}(s, s_0))}(\varphi(s)), \quad (2)$$

where  $\widehat{\varphi}_{h, \emptyset}(s_0) \doteq \mathcal{R}$ , and the  $\alpha$ -neighborhood of a result  $r \in \mathcal{R}$  is defined as the set of outcomes which are at least  $\alpha$ -similar to  $r$ :

$$\mathcal{N}_\alpha(r) \doteq \{r' \in \mathcal{R} \mid \sigma_{\mathcal{R}}(r, r') \geq \alpha\}. \quad (3)$$

Note that Definition 5 and the inference scheme (2) can be applied in a straightforward way to the concept of a (local)  $\mathcal{M}$ -similarity profile. Besides, the individual prediction (2) can be extended to a *case-based approximation* of  $\varphi$ :

$$\widehat{\varphi}_{h, \mathcal{M}} : \mathcal{S} \rightarrow 2^{\mathcal{R}}, s \mapsto \bigcap_{s' \in \mathcal{M}^\perp} \mathcal{N}_{h(\sigma_{\mathcal{S}}(s, s'))}(\varphi(s')). \quad (4)$$

Of course, assuming the profile of a CBI setup to be unknown, one cannot guarantee the admissibility of a hypothesis  $h$  and, hence, the correctness of (2). That is, it might happen that  $\varphi(s_0) \not\subseteq \widehat{\varphi}_{h, \mathcal{M}}(s_0)$ . In fact, we might even have  $\widehat{\varphi}_{h, \mathcal{M}}(s_0) = \emptyset$ . Nevertheless, taking for granted that  $h$  is indeed a good approximation of  $h_\Sigma$ , it seems reasonable to derive  $\widehat{\varphi}_{h, \mathcal{M}}(s_0)$  according to (2) as an approximation of  $\widehat{\varphi}_{h_\Sigma, \mathcal{M}}(s_0)$ . By quantifying the probability of correctness, our results below will provide a formal justification for this approach.

Note that the above inference scheme can be realized with any pair of similarity measures  $\sigma_{\mathcal{S}}$  and  $\sigma_{\mathcal{R}}$ . One should realize, however, that these measures essentially determine the similarity profile  $h_\Sigma$ . In fact, the more suitably they are defined, the more constraining  $h_\Sigma$  becomes. Thus, the choice of  $\sigma_{\mathcal{S}}$  and  $\sigma_{\mathcal{R}}$  has an important influence on the (potential) precision of the predictions (2).

Let us finally mention that (2) remains correct if the intersection is taken over  $k < n$  of the situations  $s \in \mathcal{M}^\perp$ . Since less similar situations will often hardly contribute to the precision of predictions, it might indeed be reasonable to proceed from  $k$  situations maximally similar to  $s_0$ , especially if the intersection of neighborhoods (3) is computationally complex.

### 3 Learning Similarity Hypotheses

The inference scheme (2) reveals that CBI can essentially be seen as an *instance-based* approach. Still, it also contains a *model-based* component, namely the similarity hypothesis  $h$ . Consequently, *learning* can be realized in (at least) two ways in CBI: By storing new cases in the memory, and by estimating the similarity profile. Here, we concentrate on the latter (model-based) aspect.

**Definition 6 (CBL).** Consider a CBI setup  $\Sigma$  with a memory  $\mathcal{M} \subset \mathcal{D} = \mathcal{D}_N = (c_1, \dots, c_N)$ , where  $\mathcal{D}$  denotes the sequence of cases which have been encountered so far. Moreover, let  $\mathcal{H}$  be a hypothesis space of functions  $h : [0, 1] \rightarrow [0, 1]$ . The task of case-based learning (CBL) is understood as deriving an optimal hypothesis  $h_* \in \mathcal{H}$  from the data given.

### 3.1 Optimal Hypotheses

Definition 6 has not commented on the criteria which decide on the optimality of hypotheses. Of course, an optimal hypothesis  $h_*$  should first of all be *consistent* with given data in the sense that (1) is satisfied for observed cases:

$$\sigma_{\mathcal{S}}(s, s') = x \Rightarrow \sigma_{\mathcal{R}}(\varphi(s), \varphi(s')) \geq h_*(x) \quad (5)$$

for all  $s, s' \in \mathcal{D}^\perp$ . However, testing (5) requires the consideration of all pairs of cases  $(c, c') \in \mathcal{D} \times \mathcal{D}$ . Thus, we have to relax the above consistency principle if we suppose that cases are encountered *successively*, and that not all of them are stored in the memory. In fact, we should assume (5) to hold true for all

$$(c, c') \in \mathcal{C} = \mathcal{C}_N \doteq \bigcup_{0 \leq n \leq N-1} \mathcal{M}_n \times (c_{n+1}),$$

where  $\mathcal{M}_n$  denotes the memory after the observation of the  $n$ th case  $c_n$ . We denote by  $\mathcal{H}_{\mathcal{C}} \subset \mathcal{H}$  the class of hypotheses which are consistent in this (weaker) sense.

Observe that  $h \equiv 0$  is always consistent, but leads to the trivial prediction  $\hat{\varphi}_{h, \mathcal{M}}(s_0) = \mathcal{R}$ . It is, therefore, reasonable to prefer *strong* hypotheses, which is in accordance with a *principle of minimum uncertainty* [11]. Indeed,  $h \leq h'$  entails  $\hat{\varphi}_{h', \mathcal{M}}(s) \subset \hat{\varphi}_{h, \mathcal{M}}(s)$  for all  $s \in \mathcal{S}$ . Thus, we call a hypothesis  $h_*$  *optimal* if  $h_* \in \mathcal{H}_{\mathcal{C}}$ , and if no hypothesis  $h \in \mathcal{H}_{\mathcal{C}}$  exists which is stronger than  $h_*$ . Let  $\mathcal{H}_*$  denote the class of optimal hypotheses.

**Observation 1** Suppose the hypothesis space  $\mathcal{H}$  to satisfy  $h \equiv 0 \in \mathcal{H}$  and  $(h, h' \in \mathcal{H}) \Rightarrow (h \vee h' \in \mathcal{H})$ , where  $h \vee h'$  is defined as  $x \mapsto \max\{h(x), h'(x)\}$ . Then, a unique optimal hypothesis  $h_* \in \mathcal{H}$  exists, and  $\mathcal{H}_{\mathcal{C}} = \{h \in \mathcal{H} \mid h \leq h_*\}$ .

Given the assumptions of Observation 1, CBL can be realized as a *Candidate-Elimination* algorithm [13], where  $h_*$  is a compact representation of the *version space*, i.e., the subset  $\mathcal{H}_{\mathcal{C}}$  of hypotheses from  $\mathcal{H}$  which are consistent with the training examples.

Observe that (5) guarantees consistency in the “empirical” sense that  $r \in \hat{\varphi}_{h, \mathcal{M}}(s)$  for all  $\langle s, r \rangle \in \mathcal{D}$ . Still, one might think of further demanding a kind of “logical” consistency, namely  $\hat{\varphi}_{h, \mathcal{M}}(s') \neq \emptyset$  for all  $s' \in \mathcal{S}$ . Of course, this additional demand would greatly increase the complexity of testing consistency. Moreover, the assumptions of Observation 1 would no longer guarantee the existence of a unique optimal hypothesis.

### 3.2 A Learning Algorithm

Let hypotheses be represented as step functions

$$h : x \mapsto \sum_{k=1}^m \beta_k \cdot 1_{A_k}(x), \quad (6)$$



where  $A_k = [\alpha_{k-1}, \alpha_k]$  for  $1 \leq k \leq m-1$ ,  $A_m = [\alpha_{m-1}, \alpha_m]$ , and  $0 = \alpha_0 < \alpha_1 < \dots < \alpha_m = 1$  defines a partition of  $[0, 1]$ . The class  $\mathcal{H}_{step}$  of functions (6), defined for a fixed partition, does obviously satisfy the assumptions of Observation 1. The optimal hypothesis  $h_*$  is defined by the values

$$\beta_k = \min_{(s, s') \in \mathcal{C}^\perp, \sigma_{\mathcal{S}}(s, s') \in A_k} \sigma_{\mathcal{R}}(\varphi(s), \varphi(s')) \quad (7)$$

for  $1 \leq k \leq m$ , where  $\min \emptyset = 1$  by convention. Now, suppose that  $\mathcal{M}$  is the current memory, and that a new case  $c_0 = \langle s_0, r_0 \rangle$  has been observed. Updating  $h_*$  can then be achieved by passing the iteration

$$\beta_{\kappa(s_0, s_j)} = \min\{\beta_{\kappa(s_0, s_j)}, \sigma_{\mathcal{R}}(r_0, r_j)\} \quad (8)$$

for  $1 \leq j \leq \text{card}(\mathcal{M})$ . The index  $1 \leq \kappa(s, s') \leq m$  is defined for situations  $s, s' \in \mathcal{S}$  by  $\kappa(s, s') = k \Leftrightarrow \sigma_{\mathcal{S}}(s, s') \in A_k$ . As (8) shows, the representation (6) is computationally very efficient. We refer to the algorithm defined by (8) as CBLA, and denote by  $\text{CBLA}(\mathcal{C})$  the hypothesis (7).

For obvious reasons we call  $h^* \in \mathcal{H}_{step}$  defined by the values

$$\beta_k^* = \inf\{h_{\Sigma}(x) \mid x \in D_{\mathcal{S}} \cap A_k\} \quad (9)$$

( $1 \leq k \leq m$ ) the *optimal admissible* hypothesis. Since admissibility (in the sense of Definition 5) implies consistency, we have  $h^* \leq h_*$ .

## 4 Properties of Case-Based Learning

In order to be able to study formal properties of CBLA we assume that situations are chosen repeatedly (and independently) according to some probability measure  $\mu_{\mathcal{S}}$  over  $\mathcal{S}$ . This kind of probabilistic setting is typically assumed in machine learning and can also be found in recent (more formally oriented) approaches to CBR.

We shall now consider an iterative scheme which is in accordance with the idea of CBR as a repeated process of problem solving and learning. This process, CBLP, is based on a random sequence  $(S_N)_{N \geq 1}$  of situations  $S_N \in \mathcal{S}$  which are distributed according to  $\mu_{\mathcal{S}}$ , and a sequence  $p = (p_N)_{N \geq 1} \in [0, 1]^\infty$ :

```

 $\mathcal{M}_0 = \emptyset, \ h_0 \equiv 1$ 
for  $N = 0, 1, 2, \dots$ 
  compute  $\hat{r}_{N+1} = \hat{\varphi}_{h_N, \mathcal{M}_N}(s_{N+1})$ 
  solve-problem( $s_{N+1}, \hat{r}_{N+1}$ )
   $h_{N+1} = \text{update}(h_N, c_{N+1}, \mathcal{M}_N)$ 
   $\mathcal{M}_{N+1} = \begin{cases} \mathcal{M}_N \cup (c_{N+1}) & \text{w.p. } p_{N+1} \\ \mathcal{M}_N & \text{w.p. } 1 - p_{N+1} \end{cases}$ 

```

Here,  $\text{update}(h_N, c_{N+1}, \mathcal{M}_N)$  denotes the hypothesis obtained from  $h_N$  by passing the iteration (8) for  $\mathcal{M}_N$  and the case  $c_{N+1} = \langle s_{N+1}, \varphi(s_{N+1}) \rangle$ . Moreover, **solve-problem** is a procedure in which the prediction  $\hat{r}_{N+1}$  is used for supporting the derivation of the true outcome  $\varphi(s_{N+1})$ . Observe that CBLP guarantees  $h_N = \text{CBLA}(\mathcal{C}_N)$ , but that we generally have  $h_N \neq \text{CBLA}(\mathcal{D}_N \times \mathcal{D}_N)$ . The probabilistic extension of the memory in CBLP (w.p. is short for “with probability”) takes into account that adding all observations to  $\mathcal{M}$ , i.e., taking  $p \equiv 1$ , might not be advisable [17]. Of course, efficient problem solving will generally assume a more sophisticated strategy for the instance-based aspect of learning, i.e., for maintaining the memory of cases. It might be reasonable, e.g., to take the “quality” of individual cases into account, and to allow for removing already stored cases from the memory [19]. Nevertheless, the probabilistic extension in CBLP allows for gaining insight into theoretical properties of the learning scheme. Observe that  $p_N = 0$  for  $N \geq N_0$  (with  $N_0$  being a constant number) comes down to using a fixed memory  $\mathcal{M}$ .

Given a CBI setup and the sequence  $(p_N)_{N \geq 1}$ , the hypotheses  $h_N$  induced by CBLP are random functions with well-defined (even though tremendously complicated) distributions. We are now going to derive some important properties of the sequence  $(h_N)_{N \geq 1}$ . (The proofs of subsequent results can be found in an extended version of this paper [8].) Of course, one of the first questions arising in connection with our learning scheme concerns the relation between  $(h_N)_{N \geq 1}$  and the optimal admissible hypothesis  $h^*$ .

**Theorem 1.** *Suppose  $p \geq \delta > 0$ , i.e.,  $p_N \geq \delta$  for all  $N \in \mathbb{N}$ , and let  $(h_N)_{N \geq 1}$  be the sequence of hypotheses induced by CBLP. Then,  $h_N \searrow h^*$  stochastically as  $N \rightarrow \infty$ . That is,  $h_N \geq h^*$  for all  $N \in \mathbb{N}$  and  $\Pr(|h_N - h^*|_\infty \geq \varepsilon) \rightarrow 0$  for all  $\varepsilon > 0$ .*

Observe that the stochastic convergence (from above) of the hypotheses  $(h_N)_{N \geq 0}$  toward  $h^* \in \mathcal{H}_{\text{step}}$ , which is guaranteed by Theorem 1, does not imply that  $h_N(x) \rightarrow h_\Sigma(x)$  for all  $x \in D_S$ . In fact, it might happen that  $(h^*|_{D_S})$  is already a poor approximation of  $h_\Sigma$  (at least in the strong sense of the  $|\cdot|_\infty$  metric). Moreover, the fact that  $(h_N)_{N \geq 0}$  converges *from above* suggests that we will generally have  $h_N(x) > h_\Sigma(x)$  for some  $x \in D_S$  in the course of CBLP. Thus, we might work with inadmissible hypotheses, which seems to conflict with the objective of providing correct predictions (2). Observe, however, that the (case-based) approximation  $\hat{\varphi}_{h_N, \mathcal{M}_N}$  is derived from a limited number of constraints (3), namely those associated with cases  $\langle s, \varphi(s) \rangle \in \mathcal{M}_N$ . Thus, the fact that  $h_N \searrow h^*$  does by no means rule out the possibility that  $\hat{\varphi}_{h_N, \mathcal{M}_N}$  yields correct predictions. In this connection, one might be interested in the probabilities

$$q_{N+1} = \Pr(\varphi(s_{N+1}) \neq \hat{\varphi}_{h_N, \mathcal{M}_N}(s_{N+1})) \quad (10)$$

( $N \in \mathbb{N}$ ) of incorrect predictions.

**Theorem 2.** *The following estimation holds true for the probability (10):*

$$q_{N+1} \leq 2m / \left( 1 + \sum_{k=1}^N p_k \right), \quad (11)$$

where  $m$  is the size of the partition underlying  $\mathcal{H}_{step}$ .

**Corollary 1.** *Suppose  $p \geq \delta > 0$ . Then,  $q_{N+1} \leq 2m/(\delta N + 1)$ . Particularly,  $q_{N+1} \leq 2m/(N + 1)$  if  $p \equiv 1$ .*

According to the above results, the probability of an incorrect prediction becomes small for large memories, even though the hypotheses  $h_N$  might be inadmissible. Under the assumptions of Corollary 1, this probability tends toward 0 with a convergence rate of order  $O(1/N)$ .

The upper bound established in Theorem 2 might suggest to reduce the probability of an incorrect prediction by reducing the size  $m$  of the partition underlying  $\mathcal{H}_{step}$ . Observe, however, that this will also lead to a less precise approximation of  $h_{\Sigma}$  and, hence, to less precise predictions of outcomes. “Merging” two neighbored intervals  $A_k$  and  $A_{k+1}$ , for instance, means to define a new hypothesis  $h$  with  $h|(A_k \cup A_{k+1}) \equiv \min\{\beta_k, \beta_{k+1}\}$ . In fact, the probability of an incorrect prediction can be made arbitrarily small by increasing the size of the memory. The precision of the predictions, however, is limited by the precision to which  $h_{\Sigma}$  can be approximated by  $h^*$  and, hence, by the granularity of the partition underlying the definition of the hypothesis space  $\mathcal{H}_{step}$ . Of course, nothing prevents us from extending the approach such that it allows for the adaptation of the partition. A refinement of the latter will make sense, e.g., if the size of the memory becomes large.

Summing up, the results of this section throw light on some interesting properties of our approach to case-based learning. In a statistical sense, this method allows for estimating *confidence regions*, where the corresponding level of confidence depends on the number of observed cases and can thus be quantified in advance. Moreover, this level can be made arbitrarily large by extending the size of the memory. Let us finally mention that similar results can also be obtained for CBI based on (local)  $\mathcal{M}$ -similarity profiles [8].

## 4.1 Examples

Due to reasons of space we refer to [8] for experimental studies. Here, we only present two small examples which are clearly not intended as an empirical validation of our method. They should merely convey a first idea of how CBL typically performs.

Consider the simple function  $\varphi : s \mapsto \sin(s + 1) \cdot \cos^2(s)$  defined on  $\mathcal{S} = [0, \pi/2] \cap \mathbb{Q}$ .<sup>3</sup> By averaging over a large number of learning processes (with  $p \equiv 1$ ,  $\sigma_{\mathcal{S}}, \sigma_{\mathcal{R}} : (a, b) \mapsto 1 - \exp(-(a - b)^2)$ ) and a hypothesis space based on a partition

<sup>3</sup>  $\mathbb{Q}$  is the set of rational numbers.

of size  $m = 20$ ) we have derived approximations of two quantities characterizing the quality of predictions: The *correctness* of the  $N$ th prediction is specified as  $1 - q_N$ , with  $q_N$  being defined according to (10). The *precision* of the  $N$ th prediction is captured by the expected value of  $\text{diam}(\hat{\varphi}_{h_{N-1}, \mathcal{M}_{N-1}}(S_N))$ , where  $\text{diam}(A)$  denotes the diameter of the set  $A$ . The simulation results are shown in Figure 1.

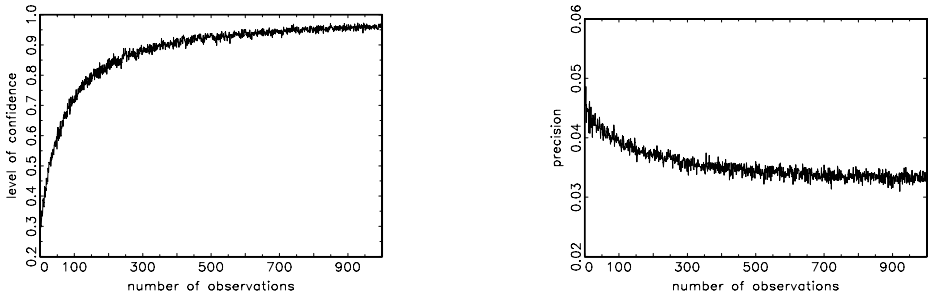


**Fig. 1.** Correctness (level of confidence) and precision for the approximation of  $\varphi : s \mapsto \sin(s + 1) \cdot \cos^2(s)$

Our example above satisfies conditions under which it can be proved that the precision converges toward 0 [8].<sup>4</sup> This is in general not the case for “real-world” applications satisfying the CBR hypothesis to a lesser degree. In connection with the HOUSING DATABASE,<sup>5</sup> for instance, we have used CBI for predicting prices of houses which are characterized by 13 attributes.<sup>6</sup> For randomly chosen memories of size 30 we have used 450 cases as training examples in order to learn the respective local  $\mathcal{M}$ -profiles. Based on (local) hypotheses thus obtained, CBI allowed for predicting prices of the remaining 56 cases with a precision of approximately 10000 dollars and a confidence level around 0.85. Taking the center of an interval as a point-estimation, one thus obtains predictions of the form  $x \pm 5000$  dollars. As can be seen, these estimations are quite reliable but not extremely precise (the average price of a house is approximately 22500 dollars). In fact, this example clearly points out the limits of an inference scheme built upon the CBR hypothesis. Our approach takes these limits into account and makes them explicit: A similarity-based prediction of prices cannot be confident and extremely precise at the same time, simply because the housing data meets the CBR hypothesis but moderately. Needless to say, problems of such type are of a general nature and by no means specific to similarity-based inference. Linear regression, for example, assumes a linear relationship between the dependent and independent variables. It yields poor predictions and imprecise confidence intervals if this assumption is not satisfied (which is seldom the case in practice).

<sup>4</sup> The convergence is often faster than in this example, but not necessarily monotone.

<sup>5</sup> Available at <http://www.ics.uci.edu/~mlearn>.

<sup>6</sup> Similarity was defined as an affine function of the distance between (real-valued) attribute values.

## 5 Concluding Remarks

We have proposed an inference scheme which is intended to support case-based problem solving by predicting a set of possible solutions to a new problem. As opposed to purely instance-based approaches, CBI makes use of an explicit model of the (otherwise implicitly used) CBR hypothesis, thereby combining instance-based and model-based learning. This way, it becomes possible to take uncertainty in CBR into account: Bearing in mind that the CBR hypothesis might apply to some problem class in a limited scope, CBI does not pretend a precision or credibility of case-based predictions which is actually not justified. At a formal level, uncertainty is expressed by supplementing (set-valued) predictions with a level of confidence.

From a statistical point of view, our method can be seen as a non-parametric approach to estimating confidence regions, which makes it also interesting for statistical inference. In [10], an instance-based prediction method has been advocated as an alternative to linear regression techniques. By deriving set-valued instead of point estimations, CBI somehow combines advantages from both methods: Like the instance-based approach it requires less structural assumptions than (parametric) statistical methods. Still, it allows for quantifying the uncertainty related to predictions by means of confidence regions.

Our main concern in this paper was aimed at the *correctness* of predictions (2). Still, it is also possible to obtain results related to the *precision* of predictions. In [8], for instance, a result similar to the one in [10] has been shown: Provided that  $\varphi$  satisfies certain continuity assumptions, it can be approximated by a function (4) (with finite memory) to any degree of accuracy. That is,  $\varphi(s) \in \hat{\varphi}_{h,\mathcal{M}}(s)$  for all  $s \in \mathcal{S}$ , and  $\text{diam}(\hat{\varphi}_{h,\mathcal{M}}) = \sup_{s \in \mathcal{S}} \text{diam}(\hat{\varphi}_{h,\mathcal{M}}(s))$  can be made arbitrarily small.

We have proposed the use of *local* similarity profiles in order to overcome the problem that globally admissible hypotheses might be too restrictive for some applications. In this connection, let us also mention a further idea of weakening the concept of globally valid similarity bounds, namely the use of *probabilistic* similarity hypotheses [9].

There are different possibilities of generalizing the method of CBL proposed in this paper. Firstly, it has already been mentioned in Section 2 that the success of CBI is essentially determined by the similarity measures  $\sigma_{\mathcal{S}}$  and  $\sigma_{\mathcal{R}}$ . It seems reasonable, therefore, to allow for the adaptation of these measures in CBL. Secondly, the instance-based aspect of learning, i.e., the organization of the memory, has to be investigated in more detail.

Let us finally mention an interesting alternative to the approach discussed in this paper, namely the idea of proceeding from a *fixed* hypothesis and of learning the probability that it yields correct predictions. More generally, this idea might be applied to a set of hypotheses (linearly ordered by  $\leq$ ) in order to obtain a related *class* of predictions. This way, not only the correctness but also the precision of predictions can be controlled.

## Acknowledgements

The author gratefully acknowledges financial support in the form of a TMR research grant funded by the European Commission.

## References

1. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994. 124
2. D. W. Aha, editor. *Lazy Learning*. Kluwer Academic Publ., 1997. 124
3. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991. 124, 125
4. B. V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991. 125
5. D. Dubois, F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, and H. Prade. Fuzzy set modelling in case-based reasoning. *Int. J. Intelligent Systems*, 13:345–373, 1998. 124
6. F. Esteva, P. Garcia, L. Godo, and R. Rodriguez. A modal account of similarity-based reasoning. *Int. J. Approximate Reasoning*, 16:235–260, 1997. 124
7. B. Faltings. Probabilistic indexing for case-based prediction. *Proceedings ICCBR-97*, pages 611–622. Springer-Verlag, 1997. 124
8. E. Hüllermeier. Similarity-based inference as constraint-based reasoning: Learning similarity hypotheses. Technical Report 64, Department of Economics, University of Paderborn, September 1999. 131, 132, 133, 134
9. E. Hüllermeier. Toward a probabilistic formalization of case-based inference. In *Proc. IJCAI-99*, pages 248–253, Stockholm, Sweden, July/August 1999. 124, 125, 134
10. D. Kibler and D. W. Aha. Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5:51–57, 1989. 125, 134
11. G. J. Klir and M. J. Wierman. *Uncertainty-Based Information*. Physica-Verlag, Heidelberg, 1998. 129
12. J. L. Kolodner. *Case-based Reasoning*. Morgan Kaufmann, San Mateo, 1993. 124
13. T. M. Michell. Version spaces: A candidate elimination approach to rule learning. In *Proceedings IJCAI-77*, pages 305–310, 1977. 129
14. E. Plaza, F. Esteva, P. Garcia, L. Godo, and R. Lopez de Mantaras. A logical approach to case-based reasoning using fuzzy similarity relations. *Journal of Information Sciences*, 106:105–122, 1998. 124
15. C. Reiser and H. Kaindl. Case-based reasoning for multi-step problems and its integration with heuristic search. *Proc. EWCBR-94*, pages 113–125, 1994. 126
16. R. Short and K. Fukunaga. The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory*, 27:622–627, 1981. 127
17. B. Smyth and P. Cunningham. The utility problem analysed. *Proc. EWCBR-96*, pages 392–399. Springer-Verlag, 1996. 131
18. B. Smyth and E. McKenna. Building compact competent case-bases. *Proc. ICCBR-99*, pages 329–342, 1999. 127
19. S. Smyth and T. Keane. Remembering to forget. *Proc. IJCAI-95*, pages 377–382, 1995. 131

# Genetic Algorithms to Optimise CBR Retrieval\*

Jacek Jarmulak<sup>1</sup>, Susan Craw<sup>1</sup>, and Ray Rowe<sup>2</sup>

<sup>1</sup> School of Computer and Mathematical Sciences  
The Robert Gordon University, Aberdeen, AB25 1HG, UK  
{jacek,s.craw}@scms.rgu.ac.uk

<sup>2</sup> AstraZeneca, Silk Road Business Park  
Macclesfield, Cheshire SK10 2NA, UK

**Abstract.** Knowledge in a case-based reasoning (CBR) system is often more extensive than simply the cases, therefore knowledge engineering may still be very demanding. This paper offers a first step towards an automated knowledge acquisition and refinement tool for non-case CBR knowledge. A data-driven approach is presented where a Genetic Algorithm learns effective feature selection for inducing case-base index, and feature weights for similarity measure for case retrieval. The optimisation can be viewed as knowledge acquisition or maintenance depending on whether knowledge is being created or refined. Optimising CBR retrieval is achieved using cases from the case-base and only minimal expert input, and so can be easily applied to an evolving case-base or a changing environment. Experiments with a real tablet formulation problem show the gains of simultaneously optimising the index and similarity measure. Provided that the available data represents the problem domain well, the optimisation has good generalisation properties and the domain knowledge extracted is comparable to expert knowledge.

## 1 Introduction

One attraction of case-based reasoning (CBR) is its ability to utilise existing data as cases. Nevertheless, constructing CBR systems for some types of problem requires significant knowledge engineering effort [7], therefore there is a need for tools that reduce the knowledge acquisition and refinement effort. In CBR systems the knowledge is distributed over a number of so-called “knowledge containers” [14]: the description language for the cases, a similarity measure used in case retrieval, adaptation knowledge – all in addition to the cases themselves. When building a CBR system, knowledge engineers will often be concerned with each of these “containers” separately. Therefore, we can approach the problem of reducing the knowledge acquisition and refinement effort in a “modular” way. Techniques can be developed separately for determining the best case description, optimising the case-base organisation and case retrieval, and acquiring and refining the adaptation knowledge. Here, we turn our attention to the task of optimising case retrieval by acquiring or refining retrieval knowledge. This is

---

\* This work is supported by EPSRC grant GR/L98015 awarded to Susan Craw.

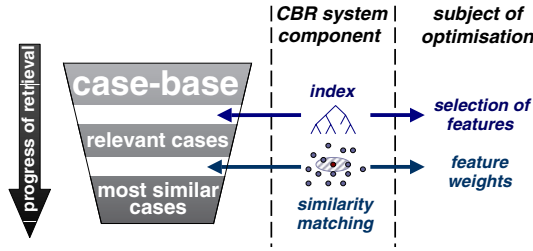


**Fig. 1.** Optimising the index and similarity measure for CBR retrieval

the first part of a larger project, that looks broadly at knowledge engineering techniques for the whole CBR process.

Figure 1 shows a retrieval scenario that is typical of many CBR systems; e.g. ReCall (ISoft)<sup>1</sup>, ReMind (Cognitive Systems Inc), Kate (AcknoSoft), and The Easy Reasoner (The Haley Enterprise). The appropriate cases are retrieved using a combined decision-tree index and  $k$ -nearest neighbour ( $k$ -NN) algorithm that applies a similarity measure. Given a new problem to be solved, the index first selects the group of cases in the leaf of the decision tree, and then a similarity measure selects those most similar to the new problem. There are two main reasons for the combined approach: an increase in speed because fewer cases are checked for similarity; and improved retrieval accuracy because fewer irrelevant cases are initially selected. CBR retrieval performance can be improved by acquiring or refining knowledge that optimises the index or the  $k$ -NN similarity measure. As Figure 1 shows, the retrieval can be optimised by learning which features should be used to induce the decision-tree index, and what weights reflect the importance of features for the similarity measure. This can be done by a domain expert, where he manually chooses the features or indicates their importance for the similarity measure, but this can be difficult.

In this paper, we concentrate on optimising the index *and* similarity measure simultaneously, where the available case data drives the optimisation. Optimisation acquires knowledge that improves the basic CBR retrieval or refines that knowledge when the case-base is re-organised or the problem-solving environment changes. Section 2 surveys related work in Machine Learning on which our approach builds. In Section 3 we introduce the real-world application on which we evaluate our techniques. The way we optimise CBR retrieval is described in Section 4. We outline the testing procedure and present experimental results in Section 5, before drawing conclusions about the achieved results in Section 6.

## 2 Related Work

Our project combines the tasks of optimising a decision-tree classifier and optimising the similarity measure for a  $k$ -NN classifier into a task of optimising

<sup>1</sup> ReCall is the tool we use and we thank ISoft for its software contribution.



CBR retrieval. Separate optimisation has been extensively treated in Machine Learning literature. For classification, sub-optimal decision trees result when induction is faced with many irrelevant features [3], even though induction itself performs feature selection, and so it is better to remove irrelevant features before induction starts [9]. Removing irrelevant features also works for neural-network classifiers [19]. The benefit is even more pronounced for nearest-neighbour classifiers [2,15]; the usefulness of decision trees as tools for feature selection for  $k$ -NN classifiers confirms this [5]. Nearest-neighbour classifiers can also be optimised by choosing feature weights for the similarity measure [16,17,11]. This is especially useful for problems where the relative importance of features plays a role in addition to (or instead of) simple feature relevance.

Optimisation algorithms form two classes: *filter* methods operate without feedback from the subsequent performance of the machine learning algorithm; *wrapper* methods utilise this feedback. Both approaches have been discussed in the context of feature selection [9,4] and feature weighting [1], with wrappers often being preferred in spite of larger computational costs [9,2,16]. Genetic Algorithm (GA) wrappers are frequently used to search for feature selections or weightings for  $k$ -NN algorithms [10,17]. Oatley *et al.* apply a GA wrapper to retrieval in a genuine CBR system [13], where the fitness is the degree of match between the ranking of the retrieved cases and that defined by the expert. However, this approach is expert intensive and was subsequently abandoned.

The *simultaneous* optimisation of the index and  $k$ -NN similarity measure is a novel contribution of this paper. The techniques above could be used to optimise the decision-tree index *or* the  $k$ -NN similarity measure, *separately*; our earlier work applied separate optimisations sequentially [8]. These experiments also showed that filter algorithms (RELIEF-F [12] and WVD [18]) did not perform well on our problem domain. Furthermore, filter methods are not suitable for optimising the index and similarity measure *built into* a CBR retrieval process, as they would ignore the bias introduced by either component. We now concentrate on a GA wrapper method to optimise the complete retrieval process.

### 3 Problem Domain

Our optimisation method has been developed for a particular application, tablet formulation, but the approach is generic and can be used for other CBR problems. Tablet formulation is a design process where, given a certain drug and its dose, a number of compounds (so-called excipients) are chosen and mixed with the drug, so that the tablet has desired mechanical properties, long-term stability, and proper drug release when swallowed. Full tablet formulation chooses a filler, binder, disintegrant, lubricant and surfactant plus quantities for each.

AstraZeneca has a rule-based system TFS that currently formulates tablets. Its development required considerable time and effort and they hoped that the availability of past formulations would entail less development effort for a CBR version. To evaluate the feasibility of CBR for tablet formulation, a hand crafted prototype, CBR-TFS-98, was developed in ReCall [6]. This experience high-

lighted the difficulty of acquiring non-case knowledge, and persuaded us to work on automated CBR knowledge acquisition, starting with case retrieval.

In this paper we present results for four subproblems of the full tablet formulation: predicting filler excipient and its amount, and predicting two tablet properties. The most difficult of the four is the filler prediction task, where we must choose an appropriate filler excipient (from 8 possible) for a tablet with a given drug and dose. We use the filler prediction task to describe the details of our optimisation approach. The filler prediction problem is represented by a feature vector containing 26 features: the dose, 5 physical properties of the drug, and 20 chemical properties of the drug with the excipients. Our domain expert tells us that some features are irrelevant for filler selection and some are derived from others.

An important aspect of tablet formulation is that several solutions may be acceptable. Therefore an *exact accuracy*, based on exactly matching the actual filler, is too restrictive as an accuracy measure during optimisation and evaluation. Instead, we constructed an approximate similarity matrix for the fillers based on the weighted Euclidean distance between the filler properties (1 boolean and 7 numerical features); the weights were provided by the expert. The expert declared that pairs of fillers with similarity greater than 0.9 should be considered as interchangeable, and those less than 0.5 were incompatible. These weights and thresholds were the only feedback on solution quality required from the expert. We used the 0.9 threshold to define the required degree of match for a weaker *correct accuracy* that was used to measure retrieval success.

## 4 Optimisation Approach

Our goal is to optimise the *complete* CBR retrieval process. Therefore, we optimise the decision-tree index so that it selects good cases for the  $k$ -NN algorithm, and the feature weighting so that it chooses the best cases from those selected. This is very different from our previous work where we combined a separately optimised decision-tree index and nearest-neighbour algorithm [8]. There, we were optimising the decision-tree index with feedback from the decision tree as a *classifier*, and optimising the similarity measure for a straight  $k$ -NN classifier. Now we provide feedback from the complete CBR retrieval stage and so undertake the simultaneous optimisation of both index and similarity. Furthermore, the optimisation of the index is no longer aimed at its classification accuracy.

We use a GA wrapper method to search the space of possible feature selections and weights. This approach is equally applicable to knowledge acquisition where the initial population is randomly seeded, and knowledge refinement where it reflects current feature selections and weights. A standard GA uses binary-valued genes to directly represent feature selections; real-valued genes for feature weights are mutated by adding a Gaussian-distributed offset. Notice that feature selections and weights are unrelated: only selected features induce the decision-tree, but all features are given similarity weights. Thus with  $F$  features and  $W$  possible weights for each feature, the  $2^F$ -dimensional space has  $2^F W^F$  vectors.

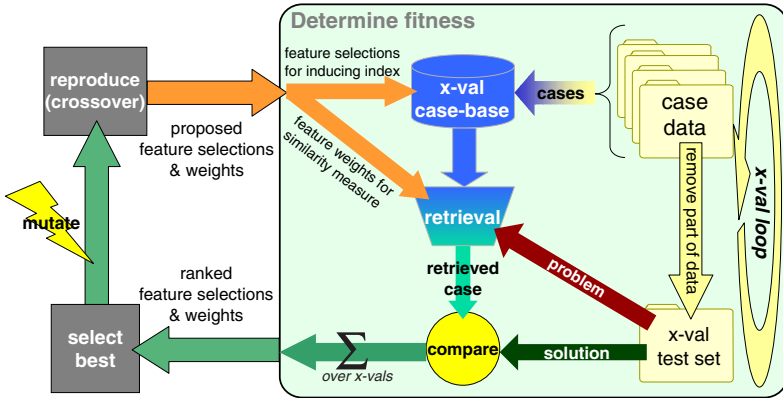


Fig. 2. Using GAs to optimise CBR retrieval

GAs are well suited to the high dimensionality of this search space and the combination of the crossover operator and selection preserves successful groups of feature selections and weights.

In each iteration of the GA, the population of feature selections and weights undergoes a process of selection, mutation and reproduction (Figure 2). New feature selections and weights replace old members of the population if their fitness is higher. The fitness function estimates the retrieval quality from the given case-base using a decision-tree induced from the selected features and applying a  $k$ -NN algorithm with the feature weights. By supplying fitness feedback from the complete CBR retrieval process we achieve simultaneous optimisation of both its components.

#### 4.1 Fitness Function

Optimisation methods can suffer from data over-fitting and so we need to incorporate cross-validation in our fitness function. We do cross-validation every time we evaluate fitness of one chromosome, the “x-val loop” in Figure 2. A leave- $n$ -drug-out cross-validation partitions the case-base repeatedly into an x-val test set containing all the case-base data for  $n$  drugs, and an x-val case-base containing the remaining drugs in the case-base. In each cross-validation experiment the decision-tree index is induced for the x-val case-base using the selected features identified by the binary-valued genes. The similarity function applies the weights identified by the real-valued genes. In this way we have defined an x-val CBR system which is now evaluated on the x-val test set. The solution predicted by the x-val CBR system is compared with the actual solution; this is known since the x-val test set is a subset of the cases from the original case-base.

The accuracy score from each x-val experiment on its x-val test set is the sum of the (weaker) correct accuracy and the average similarity of the predicted and actual fillers (see Section 3). The average over all x-val experiments returns the fitness for the given feature selections and weightings. This fitness function

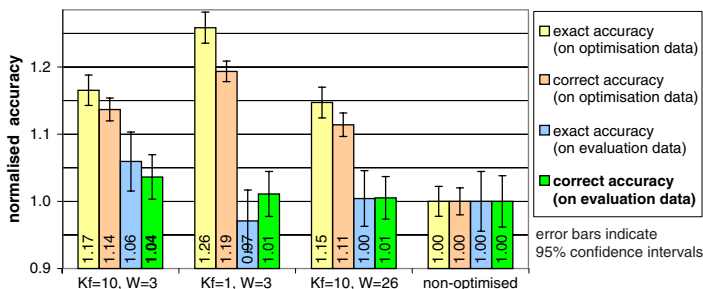


Fig. 3. Optimisation results using various optimisation parameters

provided the best results in empirical studies being smoother and taking account of finer distinctions between solutions. Experiments with other statistics of the leave-n-drug-out cross-validation, such as variance, showed no improvement.

## 4.2 Tuning the Optimisation

A popular way to achieve better generalisation in nearest-neighbour classifiers is to look at wider neighbourhoods. Following the approach in [10], we calculated fitness from the closest  $K_f$  retrieved cases ( $K_f$  may be different from the  $k$  in  $k$ -NN). Retrieved cases were ranked based on their similarity and the solution accuracies summed using rank weighting: the first has weight  $K_f/K_f$ , and the last weight is  $1/K_f$ . Using a limited number of possible weights speeds up the search, but can also increase comprehensibility and bring better generalisation [11]. Figure 3 reports results for various  $K_f$  neighbourhoods and  $W$  possible weight values evenly distributed in  $[0,1]$ . The chart shows the strict *exact accuracy* and weaker *correct accuracy* when optimised retrieval is applied to the *optimisation* data (case-base data) and an unseen *evaluation* set, see Section 5. Each is normalised with respect to non-optimised retrieval.

For a realistic and general measure of accuracy we are particularly interested in significant gains in *correct accuracy* on the evaluation set. Therefore, although  $K_f = 1$  gives better results for the optimisation data, poorer results on the evaluation data suggest that better generalisation is obtained with a larger neighbourhood. Less over-fitting is also achieved if fewer weights are used. The results in Section 5 use  $K_f = 10$  and only 3 possible weight values (0, 0.5, 1).

## 5 Experiments and Results

We have two databases with physical and chemical properties of 39 drugs and 20 excipients. Using these and the TFS rule-based system, we generated tablet formulations for all 39 drugs for 25 dosages ranging from 1mg to 720mg. This complete dataset contains many artificial formulations since only a relatively small number correspond to actual tablets. We use it as a source of formulations from which to create more realistic datasets with which to experiment; i.e. fewer

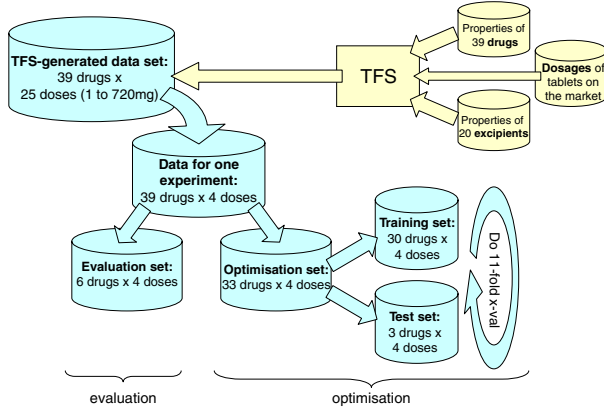


**Fig. 4.** Test procedure: data sets used

doses for each drug. For each of the 39 drugs, we randomly select a dose, then choose up to 3 more close doses, and these formulations are placed in the dataset (top half of Figure 4). If a dataset of actual formulations were available then we would use it for experiments, but we feel that these smaller datasets of 156 formulations are a good compromise since they use a more realistic selection of doses for each drug.

To determine whether the optimisation will be useful in practice, when the final CBR-TFS system solves new formulation problems, we estimate how the results of optimisation generalise to new, unseen data, by using a completely separate evaluation set. The lower half of Figure 4 shows the dataset of 156 formulations for 39 drugs being randomly split into an *optimisation* set containing formulations for 33 drugs, and an *evaluation* set containing the formulations for 6 drugs. The optimisation set provides the cross-validation training sets (x-val case-bases) and x-val test sets that generate the GA fitness. After optimisation, a CBR system is constructed where the whole optimisation set is the case-base, its decision-tree index is induced using the selected features, and its similarity measure incorporates the optimised weights. We are particularly interested in the improved performance generally and so our further discussion of effectiveness centres on the *correct accuracy* results of the retrieval by this CBR system on the *evaluation* set.

This data selection, optimisation, and evaluation procedure was repeated 40 times. In each experiment the GA population contained 100 individuals. We ran the GA through 50 cycles. Each cycle consisted of a mutation phase where 75 “mutants” were added to the population, and a crossover phase which added 90 “children”, before selection reduced the population back to 100. The time for each of the 50-cycle experiments was about 40 minutes on 400MHz SPARC. We consider this to be acceptable since the optimization will be performed off-line and typically only once. We note that we made no attempt to optimize the GA itself. In our experiments we have used 1-NN retrieval, as we found that using  $k$

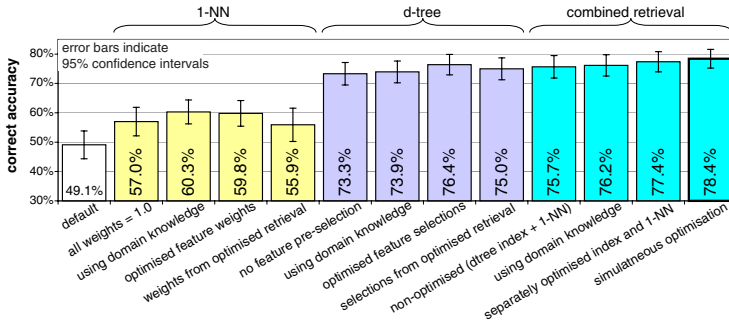


**Fig. 5.** Accuracy of filler prediction

greater than 1 gave no improvement in accuracy. We induced decision trees using `-m5` and `-c5` as C4.5's parameters, because we found that using other values of these parameters had little influence on the results.

Figure 5 compares the retrieval accuracy of several algorithms to show the effect of optimisation on the complete CBR retrieval and its individual components. The *default* algorithm, predicting the most common filler in the optimisation set, is used as a benchmark for the retrieval algorithms. *1-NN algorithms* apply a nearest-neighbour algorithm with a weighted similarity measure. The simplest algorithm has equally weighted features. The domain knowledge version applies feature weights provided by the expert; this proved to be a difficult task with the expert eventually using a binary weighting. The optimised feature weights algorithm is a separately optimised 1-NN classifier. The last algorithm takes the feature weights from the simultaneous optimisation. *d-tree algorithms* induce decision tree classifiers from various feature selections. The simplest algorithm incorporates no feature pre-selection. The domain knowledge version uses the features selected as relevant by the expert. The optimised feature selections algorithm is a separately optimised decision tree classifier. The final algorithm takes the feature selection from the simultaneous optimisation. *Combined retrieval algorithms* apply a decision-tree index followed by 1-NN matching. The simplest retrieval combines the decision-tree index with no feature pre-selection and 1-NN with equally weighted features. The domain knowledge version applies the feature selection and weighting given by the expert. The separately optimised algorithm combines the decision-tree index whose feature selection is optimised as a decision tree classifier, with 1-NN whose feature weighting is separately optimised as a 1-NN classifier. The final algorithm has feature selections and weights optimised simultaneously, where feedback is provided by the retrievals of the combined decision-tree index and 1-NN algorithm.

## 5.1 Results for Non-optimised Algorithms

Figure 5 shows the average correct accuracies on the evaluation sets. Default shows that one filler (Lactose) is dominant and accounts for almost 50% of the

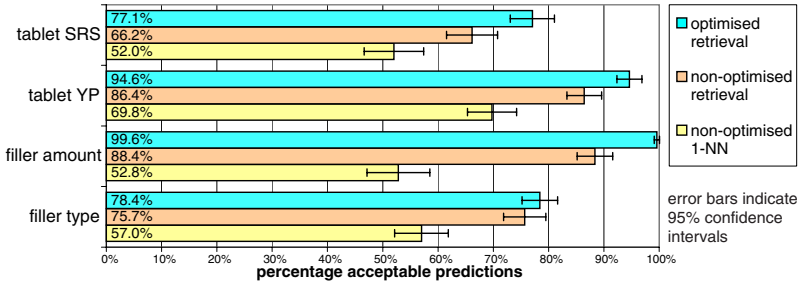


**Fig. 6.** Optimising retrieval – results for other tasks

unseen formulations. In this problem domain 1-NN accuracy is only slightly better than default filler prediction, but the decision-tree classifier achieves significantly better scores. We think this is due to the sparse nature of our data and a large number of irrelevant features. Decision-trees capture global patterns in data but similarity looks at data locally; with sparse data even the nearest neighbours can be quite different from the current problem. Therefore, in our problem domain the index plays a useful role in CBR retrieval. Although CBR systems routinely combine decision-tree indexes with NN classifiers it is valuable to see the usefulness of the index confirmed. The combined retrieval is slightly, but often not significantly, better than the corresponding decision-tree classifier.

### 5.2 Optimisation Results

The gains from optimisation in Figure 5 are not significant, but the hand-crafted knowledge from the expert also gives no significant improvement. This evidence suggests the difficulty of the formulation task and the already good results from non-optimised CBR. Optimising the weights for 1-NN retrieval achieves results comparable to using the weights selected by the expert. For the decision tree the optimised feature selections perform slightly better than expert selections. Combining the optimised index and optimised similarity measure further improves the results slightly; the best results are obtained when the index and similarity are optimised simultaneously.

It is interesting to note that when the feature selections and weights found by simultaneous optimisation are used in a simple decision-tree classifier or 1-NN classifier, neither achieve the accuracy of separate optimisation. This underlines the difference between separate and simultaneous optimisation. The best decision tree for an index may be different from that for a classifier. Similarly, different weights may be needed for similarity matching after an index, since the similarity measure is applied to only a selection of cases.

Although optimised CBR retrieval achieved better results than non-optimised methods, the gains for filler prediction were not statistically significant. We believe that it is due to the complexity of the filler-prediction problem and the sparsity of the data. Let us now consider other, less demanding, prediction tasks

**Table 1.** Match of optimisation results and domain knowledge

| task          | percentage selected features |            | average feature weight |            |
|---------------|------------------------------|------------|------------------------|------------|
|               | relevant                     | irrelevant | relevant               | irrelevant |
| filler        | 31.8%                        | 19.2%      | 0.429                  | 0.343      |
| filler amount | 72.5%                        | 5.0%       | 1.000                  | 0.005      |
| tablet YP     | 41.9%                        | 14.9%      | 0.384                  | 0.050      |
| tablet SRS    | 38.4%                        | 12.7%      | 0.534                  | 0.082      |

required in the full tablet formulation task. Predicting filler amount and some *tablet* properties are conceptually simpler problems than filler prediction. Figure 6 compares some optimisation results for filler prediction with three new tasks: predicting filler amount and two tablet properties, YP and SRS. Accuracy is again measured on the evaluation set, and the expert provided acceptability thresholds for solutions for the new tasks. Optimisation of the three new tasks brings significant improvement in retrieval. Non-optimised 1-NN retrieval is included for comparison, and we note that the large gain from using the index holds for all four tasks.

### 5.3 Correspondence to Domain Knowledge

It is important to ask how the optimised feature selections and weights correspond to those suggested by the expert. Features known to be relatively important get more higher weights, and relevant features are more often selected for the index. Table 1 shows how the optimisation results match the domain knowledge: the percentage of relevant features selected by optimisation for the decision-tree; the percentage of irrelevant features that were selected; and the average similarity weighting for both relevant and irrelevant features. Ideally we want no irrelevant features selected, and the weights for irrelevant features to be zero. However, with dependencies among features we do not necessarily need all relevant features to be selected. Furthermore, relevant features need not attract a weight of 1, instead non-zero weights can indicate their importance. Our results reflect the difficulty of the 4 problem-solving tasks. Predicting filler amount or tablet YP and SRS are each less complex than filler prediction, and the system successfully identifies the domain knowledge. For filler prediction, we conclude that our dataset is too small to capture the regularities from which the optimisation algorithm could clearly identify the relevant features. However, this size of dataset (156 formulations) is generous compared to real formulation.

## 6 Discussion and Conclusions

We have presented a general approach to optimise CBR retrieval where the case-base data is used for cross-validation experiments that provide fitness for a GA wrapper. CBR retrieval is optimised by selecting features from which to induce



a case-base index, and choosing feature weights for the similarity measure. This optimisation was found to improve retrieval results for complete CBR retrieval and its components. We have also observed that simultaneous optimisation of the index and similarity measure gives better results than separate optimisation, and that the resulting indexes and similarity measures are *different*.

Our tablet formulation domain has certain characteristics that make CBR and its optimisation particularly difficult: the data is quite complex and there are relatively few examples. Although for the difficult task of filler prediction the improvements were not significant, applying the same optimisation techniques to other problems did yield significant improvements. A further observation is that in problems with many irrelevant features using a decision-tree index significantly improves the retrieval accuracy compared to just  $k$ -NN retrieval.

As in other similar optimisation approaches, attention had to be paid to data over-fitting. We strove to limit the danger of over-fitting by using a cross-validation procedure within the GA fitness calculation. Looking at wider neighbourhoods ( $K_f > 1$ ) when calculating the fitness also reduces over-fitting, even when the finally optimised retrieval will incorporate single-best-match retrieval. In our domain, where different formulations can be acceptable, we found that calculating fitness using a finer evaluation of retrieval results than just counting exact matches helps to achieve better results on the unseen evaluation set.

Our goal is to reduce knowledge engineering effort, and so it is important to consider the expert costs of the optimisation process itself, as well as the knowledge gained. For our task relatively little input was required from the expert; he provided only some general feedback about the similarity of solutions. The knowledge gained during optimisation depends on whether there is sufficient case-base data to represent the domain knowledge. The sparseness of the data for the difficult task of filler prediction made optimisation more difficult, but for the other three prediction tasks we studied, optimisation successfully extracted that knowledge from the data.

The optimisation acquires retrieval knowledge to improve the basic retrieval of the CBR tool when applied during system development. It also refines existing retrieval knowledge when the case-data changes or grows, or when the CBR system must be adapted in reaction to a changing problem-solving environment. The task of filler prediction is not satisfactorily solved by a retrieval only CBR, even when optimised. Therefore, as we have always known, the CBR adaptation stage is vital. Our future work on the tablet formulation problem will now focus on the task of learning and refining adaptation knowledge.

## References

1. D. Aha. Feature weighting for lazy learning algorithms. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Norwell MA: Kluwer, 1998. 138
2. D. W. Aha and R. L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning*, pages 106–112. AAAI Press, Seattle, 1994. 138

3. H. Almuallim and T. G. Dietterich. Efficient algorithms for identifying relevant features. In *Proceedings of the Ninth Conference on Artificial Intelligence*, pages 38–45. Morgan Kaufman, Vancouver, 1992. 138
4. A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997. 138
5. C. Cardie. Using decision trees to improve case-based learning. In *Proceedings of the 10th ICML*, pages 25–32. Morgan Kaufmann, 1993. 138
6. S. Craw, N. Wiratunga, and R. Rowe. Case-based design for tablet formulation. In *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, pages 358–369, Dublin, Eire, 1998. Springer. 138
7. P. Cunningham and A. Bonzano. Knowledge engineering issues in developing a case-based reasoning application. *Knowledge-Based Systems*, 12, 1999. 136
8. J. Jarmulak and S. Craw. Genetic algorithms for feature selection and weighting. In S. S. Anand, A. Aamodt, and D. W. Aha, editors, *IJCAI-99 Workshop on Automating the Construction of Case-Based Reasoners*, pages 28–33, 1999. 138, 139
9. G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the 11th International Conference*, pages 121–129. Morgan Kaufmann, 1994. 138
10. J. D. Kelly and L. Davis. A hybrid genetic algorithm for classification. In *Proceedings of the 12th IJCAI*, pages 645–650, Sidney, Australia, 1991. 138, 141
11. R. Kohavi, P. Langley, and Y. Yun. The utility of feature weighting in nearest-neighbor algorithms. In *Proceedings of the European Conference on Machine Learning (ECML-97)*, 1997. 138, 141
12. I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proceedings of the European Conference on Machine Learning (ECML-94)*, Catania, Italy, 1994. 138
13. G. Oatley, J. Tait, and J. MacIntyre. A case-based reasoning tool for vibration analysis. In R. Milne, A. Macintosh, and M. Bramer, editors, *Applications and Innovations in Expert Systems VI: Proceedings of the BCS Expert Systems '98 Conference*, pages 132–146, Cambridge, December 1998, 1998. Springer-Verlag. 138
14. M. M. Richter. Introduction. In M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess, editors, *Case-Based Reasoning Technology: From Foundations to Applications*, Lecture Notes in Artificial Intelligence 1400. Springer Verlag, 1998. 136
15. D. B. Skalak. Prototype and feature selection by sampling and random mutation hill-climbing algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 293–301, New Brunswick, New Jersey, 1994. 138
16. D. Wettchereck and D. W. Aha. Weighting features. In *Proceedings of the 1st International Conference on CBR (ICCB-95)*, pages 347–358, 1995. 138
17. D. R. Wilson and T. R. Martinez. Instance-based learning with genetically derived attribute weights. In *Proceedings of the International Conference on Artificial Intelligence, Expert Systems, and Neural Networks (AIE'96)*, pages 11–14, 1996. 138
18. D. R. Wilson and T. R. Martinez. Improved heterogenous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997. 138
19. J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In Motoda and Liu, editors, *Feature Extraction, Construction and Selection - A Data Mining Perspective*. Kluwer, 1998. 138

# An Unsupervised Bayesian Distance Measure

Petri Kontkanen, Jussi Lahtinen, Petri Myllymäki, and Henry Tirri

Complex Systems Computation Group (CoSCo)  
P.O.Box 26, Department of Computer Science  
FIN-00014 University of Helsinki, Finland  
<http://www.cs.helsinki.fi/research/cosco/>

**Abstract.** We introduce a distance measure based on the idea that two vectors are considered similar if they lead to similar predictive probability distributions. The suggested approach avoids the scaling problem inherent to many alternative techniques as the method automatically transforms the original attribute space to a probability space where all the numbers lie between 0 and 1. The method is also flexible in the sense that it allows different attribute types (discrete or continuous) in the same consistent framework. To study the validity of the suggested measure, we ran a series of experiments with publicly available data sets. The empirical results demonstrate that the unsupervised distance measure is sensible in the sense that it can be used for discovering the hidden clustering structure of the data.

## 1 Introduction

Machine learning techniques usually aim at compressing available sample data into more compact representations called *models*. These models can then be used for solving different explorative (data mining) or predictive inference tasks. As opposed to traditional model-based machine learning, *lazy learning* [2] refers to methods that defer all essential computation (including the compression of sample data into a model or models) until the specific prediction task is completely determined. This type of predictive inference is also known as *case-based* or *instance-based reasoning* [1,3,15,22,25], reflecting the fact that the methods use a set of sample instances — the case base — in a central role in the predictive inference process.

In the model-based approach predictive inference problems are solved by building from sample data (and possibly domain-specific prior information) a mathematical model with a set of parameters, which then implements a function that can be used for solving the problem at hand. In the lazy learning approach, on the other hand, no such model is explicitly defined, but prediction problems are solved by comparing the input vector against the samples available in the case base. The actual prediction is typically done by using some *distance function* [15], so that the vectors closest to the input vector affect the result most. This means that from the predictive accuracy point of view, determining the distance function is the key issue in building CBR systems. However, defining a good distance function is difficult, as one has to solve problems related to

attribute scaling, handling of different attribute types, and so on. We argue that probability theory offers a general framework where many of these problems disappear, while the approach offers at the same time a theoretically solid approach for building accurate predictive models. In this paper we discuss alternative ways for linking the probabilistic predictive models into the lazy learning framework. To be able to do this, we must first formalize lazy learning as a concept.

Although the lazy learning scenario appears at first sight to be model-free, it is important to realize that this view is flawed: all distance functions are implicitly based on a model (a set of assumptions) that determine how the distance between two high-dimensional vectors is defined. For instance, it is easy to see that the Euclidean distance is based on an underlying probabilistic model with normally distributed, independent variables, while the Mahalanobis distance (see, e.g., [7]) assumes the multivariate normal model. Consequently, lazy-learning cannot be simply defined as the “model-free” approach to machine learning.

In [18] we made an attempt towards a formal definition of lazy learning, and suggested that if the basic idea is to refrain from using a single model (or a set of models), mathematically this corresponds to a requirement for marginalizing (integrating) over all the possible model families and individual models within such families. This type of integration implicitly suggests the lazy learning type of an approach. To see this, note that in the model-based approach the sample data can be compressed into a set of numbers called sufficient statistics [4], and these numbers are all that is required for determining the model parameters (hence the phrase “sufficient”). In the lazy learning approach, on the other hand, the integration over several model families means that a single set of sufficient statistics is not enough: the sufficient statistics with respect to a single family of models may not be sufficient with respect to some other family. This suggests that we should define lazy learning as a set of methods for which the minimal sufficient statistics is the data itself; hence all lazy learning methods are inherently based on using the sample data as it is, in other words, “case-based” by nature.

Inspired by this general theoretical framework, in [19] we suggested a novel distance measure for case-based reasoning tasks. However, although lazy or model-free in its general form, for computational reasons the system was implemented by using a single model as the basis for the suggested distance measure. What is more, the metric was mainly planned for a particular subtask in case-based reasoning, case-retrieval, and was not symmetric in nature. Furthermore, the approach was based on point probabilities of different events while it is clear that similarity should actually be based on properties of probability distributions, not individual probabilities.

In [16] we proposed and analyzed a supervised, probabilistic model-based data reduction scheme where two vectors are considered similar if they lead to similar predictive distributions, when the corresponding attribute-value pairs are given as input to the same probabilistic model. Unlike the metric in [19], this similarity metric is symmetric and is hence suitable for example for visualizing

high-dimensional data, as demonstrated in [16]. However, as can be seen in Section 2, the metric is inherently supervised in nature, and cannot be applied in unsupervised domains. What is more, the method is essentially based on a single probabilistic model, and is hence not case-based in the sense discussed above.

To overcome these limitations, in Section 3 we introduce a novel, unsupervised Bayesian distance measure based on an extension of the earlier supervised approach. In order to validate our approach, empirical results obtained with this distance measure are reported in Section 4. In the experiments reported here, the unsupervised distance measure was computed by using a “pool” of supervised Naive Bayes classifier models. The validity of the suggested distance measure was evaluated by both a set of subjective visual tests, and a set of simple crossvalidation experiments.

## 2 A Supervised Bayesian Distance Metric

Let  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  denote our case base, a collection of  $N$  vectors, and let us assume that each vector  $\mathbf{x}_i$  consists of values of  $m$  attributes  $X_1, \dots, X_m$ . In the sequel we will assume the attributes  $X_i$  to be discrete (or discretized) — extending the approach to continuous attributes is left as a goal for future work.

As discussed earlier, we argue that there exist no such thing as model-free reasoning, and we suggest that all our assumptions concerning the domain space should be explicitly listed and exploited by using formal models of the problem domain. By a model  $M$  we mean here a parametric (probabilistic) model form so that each parameterized instance  $(M, \theta)$  of the model produces a probability distribution  $P(X_1, \dots, X_m | M, \theta)$  on the space of possible data vectors  $\mathbf{x}$ . To make our presentation more concrete, for the remainder of the paper we assume that the models  $M$  represent different *Bayesian network structures* (for an introduction to Bayesian network models, see e.g., [24,23,13,6]).

The general idea suggested in [16] can be summarized as follows: *two vectors are considered similar if they lead to similar predictive distributions*, when the corresponding attribute-value pairs are given as input to the same Bayesian network model  $M$ . To make this idea more precise, we must first define the predictive distribution used in the above informal definition. In [16] this predictive distribution was determined with respect to a special *target variable*  $X_m$ , resulting in the conditional distribution

$$P(X_m \mid X_1, \dots, X_{m-1}, M). \quad (1)$$

Data vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are now considered similar if the corresponding predictive distributions are similar, i.e.

$$P(X_m \mid \mathbf{x}_i^{-m}, M) \approx P(X_m \mid \mathbf{x}_j^{-m}, M), \quad (2)$$

where  $\mathbf{x}_i^{-m}$  denotes the attribute values in vector  $\mathbf{x}_i$  without the value of the class variable  $X_m$ .

This type of similarity measures lead to supervised distance measures, and we can easily change the focus of the metric by changing the target variable  $X_m$ . The scheme is also scale invariant as we have moved from the original attribute space to the probability space where all the numbers lie between 0 and 1. This also allows us to handle different attribute types (discrete or continuous) in the same consistent framework. Furthermore, the framework fulfills the requirement stated earlier: the approach is theoretically on a solid basis as all our domain assumptions must be formalized in the model  $M$ .

The above scheme still leaves us with the question of defining a similarity measure between two predictive distributions. The standard solution for computing the distance between two distributions is to use the Kullback-Leibler divergence (see, e.g. [10]). However, this asymmetric measure is not (in its basic form) a distance metric in the geometric sense. In the empirical experiments reported in [16] it was observed that the following simpler distance metric yields good results in practice:

$$d_m(\mathbf{x}_i, \mathbf{x}_j) = 1.0 - P(\text{MAP}_m(\mathbf{x}_i) = \text{MAP}_m(\mathbf{x}_j)), \quad (3)$$

where  $\text{MAP}_m(\mathbf{x}_i)$  denotes the *maximum posterior probability* value of target variable  $X_m$  with respect to the predictive distribution (1), conditioned on the values of variables  $X_1, \dots, X_{m-1}$  in  $\mathbf{x}_i$ :

$$\text{MAP}_m(\mathbf{x}_i) = \arg \max_k P(X_m = k \mid \mathbf{x}_i^{-m}, M). \quad (4)$$

In this paper we however use a slightly different distance function  $d(\mathbf{x}_i, \mathbf{x}_j)$  based on a straightforward logarithmic transformation of the predictive probabilities:

$$d_m(\mathbf{x}_i, \mathbf{x}_j) = -\log P(\text{MAP}_m(\mathbf{x}_i) = \text{MAP}_m(\mathbf{x}_j)). \quad (5)$$

As noted in [16], extending this general approach to cases with two or more target variables is straightforward.

### 3 An Unsupervised Bayesian Distance Metric

The distance metric described in Section 2 is inherently supervised in nature, as it requires us to choose one (or more) of the domain variables to be used as the target variable. Consequently, this approach cannot be directly used for cases where no natural candidate for such a target variable exists, and we would like to process our data in a purely unsupervised manner. For such unsupervised domains, we propose the following unsupervised extension of the suggested supervised distance metric:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^m d_k(\mathbf{x}_i, \mathbf{x}_j). \quad (6)$$

Consequently, the distance between two vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$  is computed by taking each of the variables  $X_k$  in its turn as the target variable, and summing

the resulting  $m$  supervised distance measures computed by formula (5). Intuitively speaking, this means that two vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are considered similar, if the most probable outcome is the same in both cases in all the  $m$  individual supervised prediction tasks based on the conditional distributions

$$P(X_k | X_1, \dots, X_{k-1}, X_{k+1}, \dots, X_m, M), k = 1, \dots, m. \quad (7)$$

As we are using a sum of logarithms in the definition of our overall unsupervised distance function (6), this means that we are basically treating all these separate supervised prediction tasks independently.

The above scenario still leaves us with a very pragmatic and important question: what is the model structure  $M$  to be used for defining the conditional distributions (7)? Traditionally, this question is answered in the Bayesian framework by using the model  $M$  maximizing the posterior probability  $P(M | \mathbf{X})$ . Assuming the uniform prior for the model structures, this is equivalent to using the model with the highest *marginal likelihood* or *evidence*,

$$M = \arg \max_{M'} P(\mathbf{X} | M') = \arg \max_{M'} \int P(\mathbf{X} | \theta, M') P(\theta | M') d\theta. \quad (8)$$

With certain technical assumptions, this model selection criterion can be computed in closed form [8,11]. The required conditional distribution (1) can then be computed by marginalizing the joint probability distribution  $P(\mathbf{x}_i | \mathbf{X}, M)$  appropriately:

$$\begin{aligned} P(X_m = x_{im} | \mathbf{x}_i^{-m}, \mathbf{X}, M) &= \frac{P(X_m = x_{im}, \mathbf{x}_i^{-m}, \mathbf{X}, M)}{P(\mathbf{x}_i^{-m}, \mathbf{X}, M)} \\ &= \frac{P(\mathbf{x}_i | \mathbf{X}, M)}{\sum_k P(X_m = k, \mathbf{x}_i^{-m} | \mathbf{X}, M)}. \end{aligned} \quad (9)$$

We see that this approach is consistent with our definition of lazy learning in the sense that we marginalize in (8) over all the possible parameter instantiations. To follow this line of reasoning further, we see that should actually marginalize also over all the possible model structures  $M$  as well, yielding

$$P(\mathbf{x} | \mathbf{X}) = \sum_M P(\mathbf{x} | \mathbf{X}, M) P(M | \mathbf{X}) \propto \sum_M P(\mathbf{x} | \mathbf{X}, M) P(\mathbf{X} | M), \quad (10)$$

assuming a uniform prior  $P(M)$  for the model structures. Consequently, we should weight the prediction obtained by using model structure  $M$  by the marginal likelihood  $P(\mathbf{X} | M)$ .

It is obvious that computing the sum (10) in practice is not feasible as the number of possible Bayesian networks is super-exponential. The sum can be approximated efficiently if we can find the most important individual terms, i.e., model structures with high marginal likelihood. However, as noted in e.g. [12], finding high evidence model structures is an extremely difficult problem. This means that in practical situations we are dealing with model structures  $M$  that



are possibly only poor models of the “true” joint domain probability distribution, and hence some of the probabilities obtained are not correct. As demonstrated in [20], instead of trying to find good models of the joint probability distribution, in supervised classification domains it makes sense to try to find a model (or a set of models) so that the errors affect the accuracy of the conditional distribution (1) as little as possible, while we can allow the joint probability distribution to be such that the predictions concerning some other variable would be quite inaccurate.

For this reason, we suggest that instead of using a single model structure  $M$  for determining the distance measure (6), we should use  $m$  supervised models  $M_1, \dots, M_m$ , each chosen with respect to the corresponding predictive task. Unfortunately, as discussed in [12,9,20], finding accurate Bayesian network models for supervised prediction tasks is also a difficult problem. We return to this issue in Section 4 where we suggest computationally efficient heuristics for solving this problem.

## 4 Empirical Results

### 4.1 The Setup

To illustrate the validity of the suggested similarity metric, we performed a series of experiments with publicly available classification data sets from the UCI data repository [5]. The 20 data sets used and their main properties are listed in Table 1. In the preprocessing phase of the experimental setup, all continuous attributes in the data sets were discretized by using a straightforward application of the k-means algorithm. Consequently, with respect to the empirical study reported here, all the data sets were discrete.

When computing the pairwise distances between vectors  $\mathbf{x}_i$  and  $\mathbf{x}_j$  according to Equation (6), we need to determine  $m$  predictive distributions  $P(X_m | \mathbf{x}_i^{-m}, M_m)$ . As discussed in Section 3, finding good Bayesian network models  $M_m$  for supervised prediction tasks is difficult in practice. On the other hand, as demonstrated in, for example, [26,17], the structurally simple Naive Bayes classifier performs surprisingly well in many real-world classification domains, despite the fact that the model is extremely fast to construct and use. For this reason, in this series of experiments the predictive models  $M_m$  corresponded to  $m$  Naive Bayes models, each constructed with respect to the target variable  $X_m$ . The model parameters were set to their expected values (not to their maximum probability values). These parameter values lead to a predictive distribution equivalent to that obtained by integrating over the parameter values, and as demonstrated in [21], this results in improved classification accuracy in cases where the amount of data is very small.

To illustrate the validity of the suggested unsupervised distance measure, the data vectors in each data set were visualized in two-dimensional space by using a technique called *Sammon’s mapping* (see e.g. [14]). In this method the data vectors are placed in low-dimensional space in such a manner that the pairwise



**Table 1.** The data sets used in the experiments

| Dataset                      | size | #attrs. | #classes |
|------------------------------|------|---------|----------|
| Australian Credit            | 690  | 15      | 2        |
| Breast Cancer (Wisconsin)    | 699  | 11      | 2        |
| Breast Cancer                | 286  | 10      | 2        |
| Credit Screening             | 690  | 16      | 2        |
| Pima Indians Diabetes        | 768  | 9       | 2        |
| German Credit                | 1000 | 21      | 2        |
| Heart Disease (Cleveland)    | 303  | 14      | 5        |
| Heart Disease (Hungarian)    | 294  | 14      | 2        |
| Heart Disease (Statlog)      | 270  | 14      | 2        |
| Hepatitis                    | 155  | 20      | 2        |
| Ionosphere                   | 351  | 35      | 2        |
| Iris Plant                   | 150  | 5       | 3        |
| Liver Disorders              | 345  | 7       | 2        |
| Lymphography                 | 148  | 19      | 4        |
| Mole Fever                   | 425  | 33      | 2        |
| Postoperative Patient        | 90   | 9       | 3        |
| Thyroid Disease              | 215  | 6       | 3        |
| Vehicle Silhouettes          | 846  | 19      | 4        |
| Congressional Voting Records | 435  | 17      | 2        |
| Wine Recognition             | 178  | 14      | 3        |

distances in this space match as closely as possible to the pairwise distances in the original high-dimensional space. The objective function, corresponding to the relative error caused by the dimensionality reduction, is typically quite complex and finding the optimal visualization in this sense is not possible in practice, so we are left with approximative solutions. However, it should be pointed out that in this visualization context it is not necessary to aim at the absolutely optimal solution — for visualization purposes a reasonable approximation is usually quite sufficient. How to find effectively good approximations of the optimal visualization is, however, a challenging research problem on its own, and is not discussed in detail here. In the experiments reported here we used a simple iterative stochastic greedy algorithm where at each step the visual locations of two randomly chosen data vectors are optimized along the connecting line so that the objective function is optimized locally.

The practical relevance of a visualization can be indirectly measured through a data mining process, where domain experts try to capture interesting regularities from the visual image. In our case, however, no such domain experts were available, so we had to evaluate our visualization technique in a different manner. In this set of experiments, this was done by assuming that the clustering provided by the class labels in the UCI data sets is a reasonable clustering, in the sense that this clustering can be regarded as something that we should come up with, had we not seen the class labels originally. Following this line of rea-

soning, each classification data set was first pruned by removing the class labels, i.e., the column containing the values of the class variable  $X_m$ . The remaining data  $\mathbf{X}$  was then visualized in two-dimensional space by using the unsupervised approach presented here. Finally, the produced visual images were colored according to the class labels that were not used at all in the visualization process. If the resulting image was visually pleasing in the sense that the different classes (different colors) were nicely separated in the picture, it can be said that we were able to recover the original clustering in a *totally unsupervised manner, without using the class label information*.

## 4.2 The Results

The empirical results show that the suggested unsupervised distance measure is sensible in the sense that it can be used for discovering the underlying cluster structure: most of the produced visual images pass the class coloring clarity test explained above. A black-and-white example of the produced visualizations can be found in Figure 1. A full library of colored 2D images can be found at URL: <http://www.cs.Helsinki.FI/research/cosco/Projects/Visual/EWCBR2000>.

There is however a possible caveat in the above experimental procedure: basically there is no a priori reason why the unsupervised visual image produced should reflect the clustering provided by the class labels, especially if the original clustering is poor from the probabilistic modeling point of view. One way to measure the “goodness” of the clustering provided by the class labels is to evaluate the predictive accuracy of the Naive Bayes model, as this model is essentially based on clustering the data according to the class variable  $X_m$ . Leave-one-out cross-validated classification results of the Naive Bayes classifier can be found in the second column of Table 2.

We can now conjecture that the above “class-color clarity test” for the resulted visual images may fail with data sets where the performance of the Naive Bayes classifier is poor. The experimental results confirm this hypothesis: in cases where the leave-one-out cross-validated classification accuracy of the NB classifier is poor in the absolute sense (as with the Liver Disorders data set), or in the relative sense with respect to the default classification accuracy (as with the Postoperative Patient data set), the class labeled colored images are somewhat blurred. Nevertheless, we would like to emphasize that this does not mean that the unsupervised visualization technique has “failed” in these cases, but that in these (relatively few) cases the somewhat artificial empirical setup used here is not practically sensible in the first place. This means that if the data would in these cases be clustered according to the visual image produced, this could result in a probabilistic model producing more accurate predictions than the Naive Bayes classifier. This interesting idea is however not studied further here.

We believe that most people agree that the produced images (with the exception of the few cases discussed above) are visually pleasing in the sense that the original classes are clearly separable in the image. This however raises the

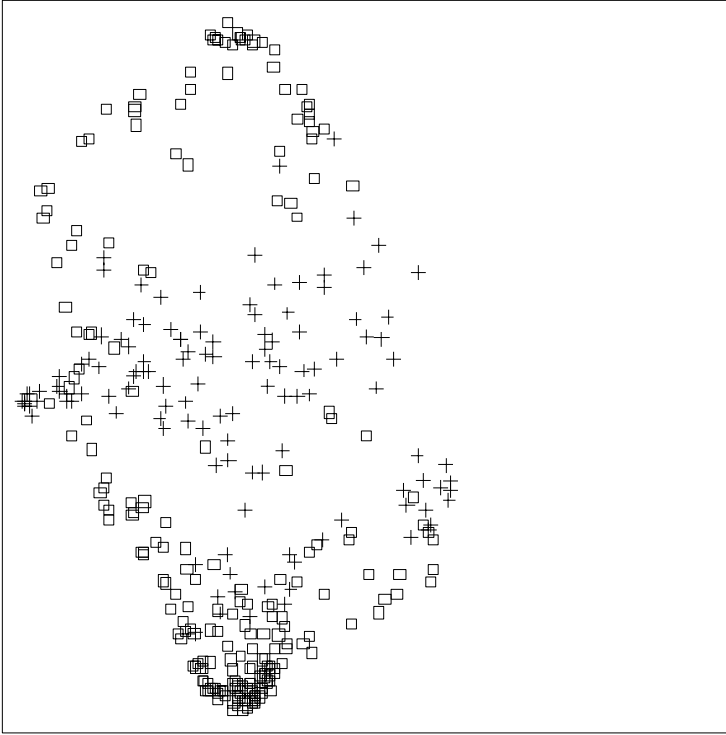


**Fig. 1.** The Ionosphere data set: an example of the unsupervised visualizations obtained with the suggested distance measure

question of whether the quality of the visualization could be measured more objectively. Intuitively, we would like to measure how well the data in the visual image is clustered according to the (hidden) class label. We suggest that this can be done by using, for example, a simple  $k$ -NN (nearest neighbor) method, where each data point is classified as a member of the class containing the most representatives in the  $k$  nearest data points. The results with this type of  $k$ -NN (with  $k=9$ ) classification method are summarized in Table 2. The method 9-NN refers to a simple 9-nearest neighbor classifier, where the distances of the  $(m-1)$ -dimensional (recall that the class labels were omitted from the data) vectors are computed by formula (6).

From Table 2 we can see that the classification accuracy obtained with the 9-NN classifier is quite comparable (and in some cases even better!) to the accuracy of the NB classifier, *even though the nearest neighbor classification was done in a purely unsupervised manner* — recall that the distance measure was computed without the class labels. This is even more surprising considering the fact that the 9-NN classifier used here was just a random and naive choice for performing CBR-type of classification, as the idea was only to demonstrate the potentiality of the suggested distance measure.

**Table 2.** Leave-one-out cross-validated classification results obtained with the data sets

| Dataset                      | Default | NB   | 9-NN |
|------------------------------|---------|------|------|
| Australian Credit            | 55.5    | 87.1 | 82.0 |
| Breast Cancer (Wisconsin)    | 65.5    | 97.4 | 97.3 |
| Breast Cancer                | 70.3    | 72.3 | 71.7 |
| Credit Screening             | 55.5    | 86.2 | 82.9 |
| Pima Indians Diabetes        | 65.1    | 77.9 | 72.7 |
| German Credit                | 70.0    | 74.9 | 67.4 |
| Heart Disease (Cleveland)    | 54.1    | 57.8 | 57.4 |
| Heart Disease (Hungarian)    | 63.9    | 83.3 | 82.7 |
| Heart Disease (Statlog)      | 55.6    | 85.2 | 83.7 |
| Hepatitis                    | 79.4    | 83.2 | 82.6 |
| Ionosphere                   | 64.1    | 92.9 | 91.2 |
| Iris Plant                   | 33.3    | 94.0 | 88.7 |
| Liver Disorders              | 58.0    | 63.2 | 58.8 |
| Lymphography                 | 54.7    | 85.8 | 81.8 |
| Mole Fever                   | 67.1    | 87.8 | 89.2 |
| Postoperative Patient        | 71.1    | 67.8 | 71.1 |
| Thyroid Disease              | 69.8    | 99.1 | 97.2 |
| Vehicle Silhouettes          | 25.8    | 64.7 | 66.5 |
| Congressional Voting Records | 61.4    | 90.1 | 88.0 |
| Wine Recognition             | 39.9    | 97.2 | 96.1 |

However, we must admit that the empirical setup used is not quite satisfactory as the two classifiers work in a different space. If one corrects this deficiency by simply incorporating the class label information into the Sammon’s mapping process, then it was observed that the 9-NN classifier outperforms the NB classifier consistently in every case. Nevertheless, it is important to notice that this type of an empirical setup is not absolutely fair either: if one incorporates all class label information in the Sammon’s mapping process, this means that when performing leave-one-out cross-validation, some information about the test vector is included in the training data, which is not acceptable. To make the comparison completely fair in every respect, the Sammon’s mapping procedure should be run  $N$  times, separately for each test vector during the loo-cross-validation cycle. This is however computationally quite demanding, and was left for a task in our future work.

Nevertheless, the results clearly suggest that the probabilistic distance-based approach used here offers a promising approach for producing accurate classifiers. This hypothesis is supported by the following argument: as described in Section 3, the unsupervised distance measure is computed by using a “pool” of  $m - 1$  Naive Bayes models, one model for each of the variables  $X_1, \dots, X_{m-1}$ , whereas the standard Naive Bayes classifier uses a single NB model, constructed with respect to the class variable  $X_m$ . As the relatively unrealistic independence assumptions made in the Naive Bayes approach do not always hold very well, the

resulting predictive distributions may be inaccurate. We conjecture that errors caused by this violation of assumptions are “smoothed” in the unsupervised case as a result of the distance measure being essentially a sum of several supervised distance measures produced by independent NB classifiers.

## 5 Conclusion

In this paper we presented an unsupervised extension of the supervised Bayesian distance metric suggested earlier. The suggested method is based on the idea that two vectors are considered similar if they lead to similar predictive probability distributions. In the earlier supervised approach this predictive distribution was defined with respect to a specific target variables (or a group of variables), whereas the suggested unsupervised metric is computed in “crossvalidated manner” by changing the target variable so that each of the variables serves in turn in this role.

The distance measure was validated in a series of experiments where the hidden structure of the data was discovered by producing a 2-dimensional visual picture of the domain. Furthermore, the distance measure was also applied in a simple nearest neighbor classifier, and the results support the visual observations: the distance measure obviously captures interesting regularities in the data. It should be emphasized that the classification results are surprisingly good with respect to those obtained by the Naive Bayes classifier, considering the fact that comparing the results is not at all fair as the nearest neighbor classifier worked in a totally unsupervised manner *without any knowledge of the class labels in the data*, whereas the Naive Bayes classifier had access to all data. This suggests that this type of an approach offers a very promising framework for building accurate case-based classifiers, but this is left as a topic for future work.

## Acknowledgements

This research has been supported by the National Technology Agency, and the Academy of Finland.

## References

1. D. Aha. *A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Observations*. PhD thesis, University of California, Irvine, 1990. 148
2. D. Aha, editor. *Lazy Learning*. Kluwer Academic Publishers, Dordrecht, 1997. Reprinted from Artificial Intelligence Review, 11:1–5. 148, 158
3. C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. In Aha [2], pages 11–73. 148
4. J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 1985. 149

5. C. Blake, E. Keogh, and C. Merz. UCI repository of machine learning databases, 1998. URL: <http://www.ics.uci.edu/~mlearn/MLRepository.html>. 153
6. E. Castillo, J. Gutiérrez, and A. Hadi. *Expert Systems and Probabilistic Network Models*. Monographs in Computer Science. Springer-Verlag, New York, NY, 1997. 150
7. C. Chatfield and A. Collins. *Introduction to Multivariate Analysis*. Chapman and Hall, New York, 1980. 149
8. G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992. 152
9. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997. 153
10. A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall, 1995. 151
11. D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, September 1995. 152
12. D. Heckerman and C. Meek. Models and selection criteria for regression and classification. In D. Geiger and P. Shenoy, editors, *Uncertainty in Artificial Intelligence 13*, pages 223–228. Morgan Kaufmann Publishers, San Mateo, CA, 1997. 152, 153
13. F. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996. 150
14. T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995. 153
15. J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, 1993. 148
16. P. Kontkanen, J. Lahtinen, P. Myllymäki, T. Silander, and H. Tirri. Using Bayesian networks for visualizing high-dimensional data. *Intelligent Data Analysis*, 2000. To appear. 149, 150, 151
17. P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. BAYDA: Software for Bayesian classification and feature selection. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 254–258. AAAI Press, Menlo Park, 1998. 153
18. P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. Bayes optimal instance-based learning. In C. Nédellec and C. Rouveirol, editors, *Machine Learning: ECML-98, Proceedings of the 10th European Conference*, volume 1398 of *Lecture Notes in Artificial Intelligence*, pages 77–88. Springer-Verlag, 1998. 149
19. P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. On Bayesian case matching. In B. Smyth and P. Cunningham, editors, *Advances in Case-Based Reasoning, Proceedings of the 4th European Workshop (EWCBR-98)*, volume 1488 of *Lecture Notes in Artificial Intelligence*, pages 13–24. Springer-Verlag, 1998. 149
20. P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. On supervised selection of Bayesian networks. In K. Laskey and H. Prade, editors, *Proceedings of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 334–342. Morgan Kaufmann Publishers, 1999. 153
21. P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, and P. Grünwald. On predictive distributions and Bayesian networks. *Statistics and Computing*, 10:39–54, 2000. 153
22. A. Moore. Acquisition of dynamic control knowledge for a robotic manipulator. In *Seventh International Machine Learning Workshop*. Morgan Kaufmann, 1990. 148
23. R. E. Neapolitan. *Probabilistic Reasoning in Expert Systems*. John Wiley & Sons, New York, NY, 1990. 150

24. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988. 150
25. C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986. 148
26. H. Tirri, P. Kontkanen, and P. Myllymäki. Probabilistic instance-based learning. In L. Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference (ICML'96)*, pages 507–515. Morgan Kaufmann Publishers, 1996. 153

# Remembering Why to Remember: Performance-Guided Case-Base Maintenance\*

David B. Leake and David C. Wilson

Computer Science Department, Indiana University  
Lindley Hall 215, Bloomington, IN 47405, U.S.A.  
{leake,davwils}@cs.indiana.edu

**Abstract.** An important focus of recent CBR research is on how to develop strategies for achieving compact, competent case-bases, as a way to improve the performance of CBR systems. However, compactness and competence are not always good predictors of performance, especially when problem distributions are non-uniform. Consequently, this paper argues for developing methods that tie case-base maintenance more directly to performance concerns. The paper begins by examining the relationship between competence and performance, discussing the goals and constraints that should guide addition and deletion of cases. It next illustrates the importance of augmenting competence-based criteria with quantitative performance-based considerations, and proposes a strategy for closely reflecting adaptation performance effects when compressing a case-base. It then presents empirical studies examining the performance tradeoffs of current methods and the benefits of applying fine-grained performance-based criteria to case-base compression, showing that performance-based methods may be especially important for task domains with non-uniform problem distributions.

## 1 Introduction

Case-base maintenance has become an active CBR research area, producing results with important ramifications for both the theory and practice of CBR. Much significant work in this area focuses on developing methods for reducing the size of the case-base while maintaining *case-base competence*, “the range of target problems that can be successfully solved” (Smyth & McKenna 1999a). Strategies have been developed for controlling case-base growth through methods such as competence-preserving deletion (Smyth & Keane 1995) and failure-driven deletion (Portinale, Torasso, & Tavano 1999), as well as for generating compact case-bases through competence-based case addition (Smyth & McKenna 1999a; Zhu & Yang 1999).

The goal of achieving compact competent case-bases addresses important performance objectives for CBR systems. First, sufficient competence is a *sine qua non* for performance: no CBR system is useful unless it can solve a sufficient

---

\* The authors’ research is supported in part by NASA under award No NCC 2-1035.



proportion of the problems that it confronts. Second, compacting the case-base may help to increase system efficiency by alleviating the utility problem for retrieval (Francis & Ram 1993; Smyth & Cunningham 1996). As an added benefit, compact case-bases decrease communications costs when case-bases are used as vehicles for knowledge sharing or are transferred in distributed CBR systems (cf. (Doyle & Cunningham 1999)).

However, case-base compactness is only a proxy for performance in a CBR system, rather than an end in itself. For example, decreased retrieval cost from a smaller case-base may be counterbalanced by increased adaptation costs or decreased quality. Thus optimizing the performance of a CBR system may require balancing tradeoffs between competence, quality, and efficiency (Portinale, Torasso, & Tavano 1999; Smyth & Cunningham 1996). In addition, adjusting the case-base to optimize performance may require reasoning about the system's performance environment, taking into account that patterns in problem distribution make some cases more useful than others (Leake & Wilson 1999). Consequently, effective maintenance requires remembering *why* cases are being remembered (or forgotten)—to serve the overall performance goals of the CBR system for a given task—and optimizing maintenance decisions accordingly. Now that research on case-base competence is becoming mature, we believe that the time is ripe to make performance criteria play a more direct role in guiding case addition and deletion.

This paper examines the benefits of using fine-grained performance metrics to directly guide case addition and deletion, and presents initial experiments on their practicality. The paper begins by discussing the competence/performance dichotomy and the factors that should guide case-base maintenance. It then illustrates the importance of adding direct performance considerations to maintenance strategies, by showing that in some cases, increased performance can be achieved without sacrificing either competence or compactness. It next presents a performance-based metric, guided by cases' contributions to adaptation performance, to guide case addition and deletion. Experiments examine the common alternative practice of reflecting performance with fixed adaptation effort thresholds, illuminating tradeoffs in adaptation cost and case-base compression, and then compare the effects of competence-based and performance-based strategies. Our results show that performance-based deletion strategies are especially promising for non-uniform problem distributions, which have received little attention in previous analyses of case-based maintenance, but which are often important in real-world contexts.

## 2 The Competence-Performance Dichotomy

Case-base maintenance is fundamentally driven by performance concerns. For example, Leake and Wilson's (1998) definition of case-base maintenance is explicitly performance-related:

Case-base maintenance implements policies for revising the contents or organization of the case-base in order to facilitate future reasoning for a particular set of performance objectives.

In this definition, the performance measure evaluates the performance of a particular CBR system for a given initial case-base and sequence of target problems.

To relate the competence and performance of CBR systems, it is useful to revisit the notions of competence and performance. When Chomsky (1965) formulated the original competence-performance dichotomy in linguistics, he used competence to describe the “in principle” abilities of an ideal speaker, unaffected by factors such as processing limitations, and used performance to describe how language was actually used by real speakers under real constraints in real situations. “Competence” in CBR has a specialized meaning—the range of target problems that a system can solve (Smyth & McKenna 1999a)—but the idea of “problems that a system *can* solve” can be taken to reflect an idealized competence. For example, if retrieval and adaptation time are allowed to be arbitrarily long, the competence of the case base for a sequence of input problems depends only on the “in principle” adequacy of system knowledge.

In practice, processing constraints are important, and current case-base competence research often reflects them in adaptation effort thresholds, which treat a case to be “adaptable” to solve a problem only if it can be adapted within a fixed limit on the number adaptation steps allowed (e.g., (Portinale, Torasso, & Tavano 1999; Zhu & Yang 1999)). Defining competence in terms of cases within the adaptation threshold combines one aspect of “idealized” competence (that the set of cases can be partitioned into adaptable and non-adaptable cases, with all adaptable cases treated as being equivalent) with the pragmatic concerns reflected in guaranteeing an upper bound on the required adaptation effort.

This paper argues for a finer-grained approach, which we call *performance-based*, to make its decisions directly reflect expected impact on top-level performance goals (in these examples, goals for processing efficiency). In order to develop this approach, we first identify the relevant performance goals and their relationships.

### 3 Performance Goals for Case-Base Maintenance

In general, there will be multiple performance measures for a CBR system, and there is no guarantee that all of them can be maximized simultaneously. In order to balance these measures to achieve the best overall performance, it is useful to distinguish top-level goals from goals that are only instrumental, rather than targets in themselves. For example, the goal of decreasing case-base size is not pursued for its own sake (provided space is available), but instead, as an instrumental goal of the higher-level goal to decrease retrieval time. Decreasing retrieval time is itself an instrumental goal to the top-level performance goal of improving problem-solving speed. A maintenance system that recognizes that compactness is an instrumental goal, rather than a top-level goal, can make

better decisions about how to manage compactness compared to other goals, for example, by sacrificing compactness when it improves performance. However, when compactness is used as a proxy for efficiency and simply maximized, the maintenance process may miss better opportunities to maximize efficiency.

Smyth and McKenna (1999) define three types of top-level goals for CBR systems:

1. Problem-solving efficiency goals (e.g., average problem-solving time)
2. Competence goals (the range of target problems solved)
3. Solution quality goals for problems solved (e.g., the error level in solutions)

Any case addition/deletion strategy must be shaped by these goals and the acceptable tradeoffs between them. In addition, we note that addition and deletion strategies are also guided by the following constraints:

1. Case-base size limits (if any)
2. Acceptable long-term/short-term performance tradeoffs
3. The availability of secondary sources of cases
4. The expected distribution of future problems

For example, Smyth and Keane's (1995) competence-preserving deletion strategies reflect all of these constraints. Their deletion process keeps the case-base within acceptable size limits (constraint 1); their competence-guided choices are intended to minimize the loss of future coverage (constraint 2); their methods' deletion choices assume a uniform distribution of problems (constraint 3); and no other sources of cases are available for recovering deleted information (constraint 4), making preservation of competence a key concern. Other instantiations of these constraints would give rise to different strategies. For example, if short-term performance is crucial and long-term is less important, and current problems are concentrated in a small part of the case-base, it may be acceptable to sacrifice current competence and build it back through future learning. By their very nature, competence criteria aim at maximizing coverage, rather than trading off coverage and efficiency based on the expected problem distribution, but as we show later in the paper, making such tradeoffs may be useful for non-uniform problem distributions.

## 4 The Value of Performance-Based Criteria

Making the right decisions about cases to retain requires augmenting competence criteria with consideration of the performance effects of alternative cases. Usually this is thought of in terms of achieving a better tradeoff between competence and efficiency. However, in some situations, performance considerations can even improve efficiency *without loss of competence or compactness*. We illustrate this with a simple example.

For this example, we assume the most easily adaptable case is always retrieved for each problem, and that the case-base is built by from a set of candidates by a greedy algorithm which, for each step, adds the candidate case

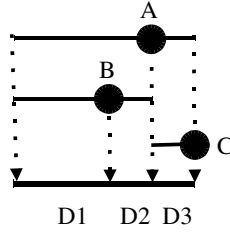


**Fig. 1.** Three example cases and their coverage

that provides the greatest increment to competence, until achieving full coverage (Zhu & Yang 1999). Consider building a case-base from 3 cases, A, B, and C as shown in Figure 1. The line segment at the bottom of the figure represents the problem space, where problems are associated with points on the line. (For example, problems could be the desired yield strength for a metal, and solutions the manufacturing processes to obtain it.) Suppose that if case  $C_1$  solves problem  $p_1$ , the cost to adapt  $C_1$  to solve a new problem  $p_2$  is  $\alpha|p_1 - p_2|$ , for some fixed  $\alpha > 0$ .

The horizontal positioning of A, B, and C along the problem axis reflects the specific problems that each one solves, and the horizontal intervals adjacent to each case reflect the space of problems that it can be adapted to solve, given the system’s adaptation knowledge. The interval surrounding A is an open interval on the right; case A cannot be adapted to solve the problem solved by case C. All other endpoints are closed. To build the case-base, a greedy competence-based case addition algorithm selects first case A first and then case C, resulting in the case-base  $CB_1 = \{A, C\}$ , which provides maximal competence. We note, that  $CB_2 = \{B, C\}$  provides the same competence.

If the problem distribution is uniform, it can be shown that the difference between the expected adaptation cost for solving problems using case-base  $CB_1$  instead of  $CB_2$  is  $\alpha D_2(D_1 - D_2/4)/(D_1 + D_2 + D_3)$ . If we fix  $D_2$  and  $D_3$  and let  $D_1 \rightarrow \infty$ , the expected average adaptation cost difference goes  $D_2$ . (Intuitively, almost all problems will then fall to the left of case B, and those problems will be  $D_2$  closer to case B than to case A.) Thus for this example, there are two competing case-bases with the same competence and the same size, but with different performance, so it is only possible to choose between them based on performance, not competence or compactness—and in fact, a competence-based greedy case addition algorithm picks the wrong one. This example demonstrates that performance-based considerations, distinct from competence and compactness, can play an important role in case-base selection.

## 5 A Performance-Based Metric for Case Selection

This section describes a strategy for performance-based case selection, inspired by Smyth and McKenna’s (1999a) RC-CNN algorithm. That algorithm compacts

case-bases using a compressed-nearest-neighbor (CNN) algorithm (Hart 1968) whose inputs are ordered by a relative coverage (RC) metric, to give priority to cases expected to make the largest competence contributions. By analogy to the RC metric, which estimates each case’s unique contribution to the competence of the system, we have developed a relative performance (RP) metric aimed at assessing the contribution of a case to the *adaptation performance* of the system.

Our RP metric depends on two standard definitions from case-base competence research, the coverage set of a case (the set of problems from the target set that the problem solves) and the reachability set of a problem (the set of cases that solve that problem). It also depends on the representativeness assumption that the contents of the case-base are a good approximation of the problems the system will encounter (see (Smyth & McKenna 1999a) for full definitions and discussion), but can be weighted to reflect different expected problem frequencies.

The RP value for a case reflects how its contribution to adaptation performance compares to that of other cases. To approximate the benefit of adding the case to the case-base, we first assume that the similarity metric will accurately select the most adaptable case for any problem. For each case that might be added to the case-base, we estimate its contribution to adaptation performance. We have explored a number of metrics, including a “performance benefit” (PB) metric estimating the actual numerical savings that the addition of each case provides. However, best results were obtained by considering a case’s relative adaptation performance, the percent savings it provides compared to the worst alternative case that solves the problem. If we let  $RS(c', c)$  stand for  $ReachabilitySet(c') - \{c\}$ , for a fixed case-base CB we define:

$$RP(c) = \sum_{c' \in CoverageSet(c)} 1 - \frac{AdaptCost(c, c')}{\max_{c'' \in RS(c', c)} AdaptCost(c'', c')}$$

This metric can be used to guide either case addition—favoring cases with high RP values—or case deletion—favoring cases with low RP values. By adding an additional weighting factor, reflecting the expected probability of new problems similar to those in the case-base being encountered in the input stream, this formula can reflect expected problem distributions. Even if the distribution is not known completely, this adjustment can refine case selection to improve performance for likely “hot spots” in the case-base (Leake & Wilson 1999).

Because the actual relative performance of a particular case depends on the other cases in the case-base, using completely accurate RP values to guide case deletion would require recalculating RP values after additions or deletions, which could be extremely expensive. A more practical alternative, which we will refer to as RP-CNN, is to do a one-time RP calculation, and then to use that estimate to order the cases presented to CNN, analogously to RC-CNN. A key question is whether this approximate information is sufficiently accurate to improve performance. We test RP-CNN and compare its effects to RC-CNN in Section 6.3.

## 6 Experimental Results

To explore the relationships between compactification strategies and performance, we conducted four experiments. These examine (1) how the choice of adaptability thresholds affects system performance, (2) the tradeoffs between compressed case-base size and expected adaptation costs for CNN, (3) the performance obtained by RC-CNN compared to RP-CNN for uniformly-distributed problems, and (4) their comparative performance for non-uniformly distributed problems.

The experiments were conducted in a simple path planning domain that models an inter-/intra-city transportation network. Concentrated areas of local connectivity represent cities. Paths are viewed as different modes of transport between locations; they do not correspond directly to grid lines but do reflect the grid distance between location points. Models are generated randomly, based on specifications of the number and size of the cities, the number of locations in each city, the minimum distance between cities, and the maximum number of paths connecting locations. The model generator ensures that all locations are reachable through some path from all other locations, if necessary adding paths to ensure connectivity.

The planner combines case-based planning with a generative (breadth-first) path planner to adapt cases by extending their paths. This enables natural control over the allowable adaptation, by setting a threshold on the allowed number of adaptation steps. Path cases represent the starting and ending locations, the path between them, and the path distance. Cases are retrieved based on minimizing the combined distance between the starting and ending locations in a case and new travel problem.

### 6.1 Performance Effects of Competence Coverage Thresholds

Competence-preserving addition and deletion methods must determine the competence contributions of cases, which depends on the system's ability to retrieve and adapt particular cases. As described previously, the adaptability judgment is often based on an adaptation threshold, with all cases that can be adapted within the threshold treated as equally adaptable to the problem. This blurs the adaptability differences between particular cases, sacrificing some ability to select high-performance cases. The first experiment examines affects the performance of case-bases generated by RC-CNN for different thresholds.

For each test, we generated a model consisting of 3 city areas of size 20 by 20, with 40 locations in each city. We randomly generated case-bases of sizes 1000, 750, 500, and 250 from the possible starting and ending location pairs in the model. Each case-base was then reduced in size by the RC-CNN method, and the reduced case-base was tested with 100 randomly selected probes from the model space. Each test was repeated 10 times, selecting a new initial case-set and test probes for each trial and averaging the results.

Higher threshold values increase the variance in adaptation costs for problems that a case covers, decreasing pressure to add nearby cases. Consequently, we

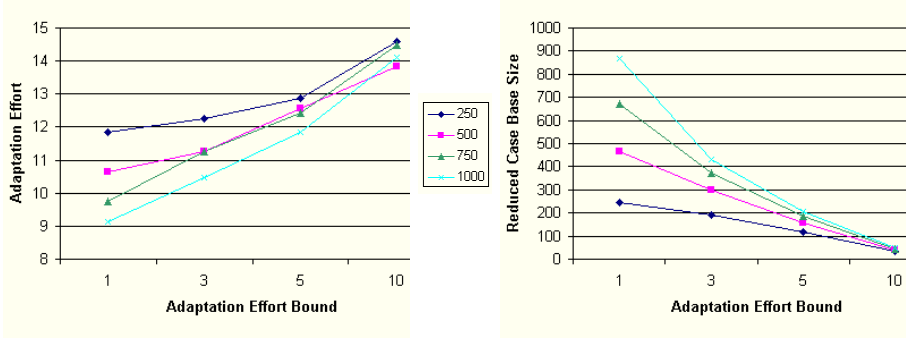


**Fig. 2.** Adaptation effort as a function of threshold (left), and reduced case-base size as a function of threshold (right), for RC-CNN compression

expected adaptation performance to decrease as the threshold values increased. This basic trend appears in the results in the left side of Figure 2, which shows average adaptation effort on the test problems as a function of the threshold. This effect is seen across all case-base starting sizes, but lower thresholds were better at exploiting the range of cases in large case bases, selecting closer cases (resulting in lower adaptation costs). Our explanation is that all the case-bases in our experiments were large enough to provide adequate coverage, but that, at high thresholds, case choice was not sufficiently selective to take full advantage of the wider choice of cases by choosing better case distributions.

## 6.2 Compressed Size vs. Adaptation Cost Tradeoffs

The previous experiment illustrates how adaptation effort thresholds used by RC-CNN can affect the adaptation effort required for a system to solve problems. However, required adaptation effort is not the only concern: There is a tradeoff because lower thresholds decrease the range of problems that each case can be used to solve, making us expect less compression to be possible for a given competence level. In this experiment we observe the effects of different case threshold levels on the case-base size obtained using RC-CNN.

Using the basic experimental procedure described previously, we determined the resulting case-base size at four different threshold levels, from 1 to 10, for initial case-bases of 250, 500, 750, and 1000 cases. We expected that as the adaptation threshold increased, the size of the case-base produced by RC-CNN would decrease. We expected that the resulting size would be ordered by the sizes of the case-bases, with the greatest compression being achieved for large case-bases. These predictions are borne out in the right side of Figure 2. It is interesting to note the very substantial compression ratio achieved for a threshold of 10.

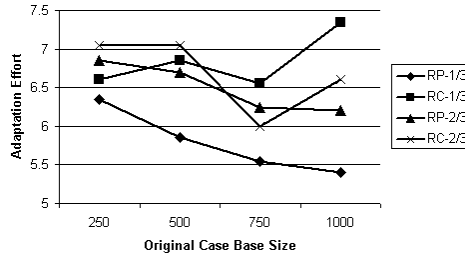


**Fig. 3.** Average adaptation effort for non-uniform case distributions

### 6.3 Comparing CNN, RC-CNN and RP-CNN for Uniform Case Distributions

A third experiment compared the effects of basic CNN, RC-CNN, and RP-CNN on case-base compression and adaptation efficiency, using the same basic procedure and starting with a case-base of size 1000, with adaptation boundary of 5. For CNN, the mean case-base size was 262, for RC-CNN, 204, and for RP-CNN, 284. With a uniform distribution of test problems, mean adaptation cost for CNN was 2.96, for RC-CNN was 3.19, and for RP-CNN was 2.87. Thus as expected, RP-CNN provided some gains in efficiency at a cost of increased case-base size, while RC-CNN provided substantial gains in case-based compression at the expense of some efficiency. This provides partial independent confirmation for the results of (Smyth & McKenna 1999a).

Although RP-CNN achieved slightly better performance than the other methods, more experiments are needed (e.g., to compare the performance achieved when the size of the resulting case-bases is held constant). We have a number of refinements in mind that we expect to improve performance for the RP metric, as well as for making the RP recalculation process more efficient, in order to be able to use more accurate RP values at each step rather than relying on a single static approximation calculated at the start of processing.

### 6.4 RC vs. RP Deletion for Non-uniform Case Distributions

In order to test the performance of our metric under non-uniform problem distributions, we designed an experiment in which routes with origin and destination in certain cities are requested more frequently. Both the number of cities that comprise the high traffic area and the frequency of requests for routes in that area are parameters of the experiment. At the beginning of the experiment, a subset of cities of the desired size is selected at random, and routes that start from and end in those cities are considered high-traffic routes. Test probes are randomly generated from the high-traffic areas in proportion to the specified frequency, with the remaining probes randomly generated from the lower-traffic areas.



Using the same model setup as in the earlier experiments, we tested conditions in which one and two of the three cities comprised the high-traffic routes. We ran the experiments with a 95 percent frequency rate for high-traffic probes, using the RP metric, with a weight factor to reflect the probability of a particular problem occurring (based simply on whether the problem was in a high-traffic area, and the probability of problems in that area). We evaluated effects on compression by case deletion, first running CNN to determine a target size for the compressed case-base, then ordering the candidate cases according to the metric being tested (RC or RP), and deleting the least desirable cases according to the metrics, until reaching the determined size. Here we expected to see greater performance benefits for RP than in the previous experiment, because RC focuses on coverage alone, while the revised RP favors useful cases in high-traffic areas. This was borne out in our results, which are shown in figure 3 for two experimental configurations: one high-traffic area and two low-traffic areas of equal size (1/3), and two high-traffic areas and one low-traffic area (2/3), for RC-CNN using an adaptation threshold of 10. The graph shows the median effort to solve cases after reduction of the case-base, for cases within the adaptation limit, for initial case-base sizes ranging from 250 to 1000 cases. For all but one test, performance with RP surpasses RC. Benefits are strongest with more focused areas (1/3), and benefits of RP appear to increase with larger initial case-bases, perhaps because the wide range of cases allows RP to fine-tune its choices.

## 7 Comparison to Previous Research

The importance of utility-based considerations for maintenance is well-known. Smyth and Keane's (1995) seminal competence work, for example, proposes footprint-utility deletion, in which case deletion decisions are based first on competence categories and then on utility. Smyth and Cunningham (1996) examine the tradeoffs between coverage, quality, and efficiency, illustrating how case-base size can affect retrieval and adaptation costs, as well as quality. van Someren, Surma, and Torasso (1997) suggest using on a cost model for the CBR system to guide decisions about the size of the case-base.

Portinale, Torasso and Tavano (1998) present a case deletion strategy aimed at favoring useful cases in a combined CBR-MBR system. Their method replaces old cases with new cases solved by the MBR system, provided the new case covers the problem of the replaced case, within a fixed adaptation effort threshold, and requires more effort than the case being replaced. The *Learning by Failure with Forgetting* strategy (Portinale, Torasso, & Tavano 1999) applies another heuristic, periodically deleting cases that have remained unused longer than a pre-defined time window and "false positive cases." These are valuable heuristic methods, but differ from the RP metric's more quantitative approach, which balances the expected future performance contributions of alternative cases in the global context of competing cases in the case-base, rather than assessing cases independently.

As discussed previously, the framework here is based on the competence modeling framework of (Smyth & McKenna 1999a). We agree with the importance of competence criteria, and plan to develop combined competence/performance metrics for tuning the maintenance process to achieve a desired balance between competence and performance concerns. For example, in a combined CBR+MBR system that can solve any problem from scratch, it may be appropriate to base maintenance decisions solely on efficiency, but in a domain where it is impossible to reconstruct deleted cases, competence concerns should receive considerable weight.

## 8 Conclusion

An important current of CBR research studies how to develop strategies for achieving case-bases that are competent and compact, as a proxy for good system performance. This paper has presented an argument for integrating performance considerations more directly into case addition and deletion procedures, in order to allow finer-grained optimization of case-base contents. The paper shows that the relationship between competence, compactness and adaptation performance is more subtle than a simple tradeoff—in some circumstances, adaptation performance can be increased without sacrificing competence or compactness—motivating the search for ways to refine case addition and deletion procedures to improve performance results. It also presents empirical studies demonstrating relationships between competence criteria, adaptation performance, and case-base size, as well as an initial step towards developing a performance-guided metric for estimating the performance value of adding a case to a case-base.

Much remains to be done in refining this approach and providing a richer model. Such work includes refining the performance metric; performing more theoretical and empirical analyses of the tradeoffs and factors involved, considering both retrieval and adaptation costs; and combining competence and performance metrics to achieve metrics that balance both factors as desired. However, we believe that just as the direct connection of retrieval criteria to adaptation abilities led to important progress (Smyth & Keane 1998), the direct connection of case-base construction to performance criteria promises important advances for case-base maintenance research.

## References

- Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA.: MIT Press.
- Doyle, M., and Cunningham, P. 1999. On balancing client-server load in intelligent web-based applications involving dialog. Technical Report TCD-CS-1999-25, Trinity College Dublin. 162
- Francis, A., and Ram, A. 1993. Computational models of the utility problem and their application to a utility analysis of case-based reasoning. In *Proceedings of the Workshop on Knowledge Compilation and Speed-Up Learning*. 162

- Hart, P. E. 1968. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory* 14:515–516. 166
- Leake, D., and Wilson, D. 1998. Case-base maintenance: Dimensions and directions. In Cunningham, P.; Smyth, B.; and Keane, M., eds., *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, 196–207. Berlin: Springer Verlag.
- Leake, D., and Wilson, D. 1999. When experience is wrong: Examining CBR for changing tasks and environments. In *Proceedings of the Third International Conference on Case-Based Reasoning*, 218–232. Berlin: Springer Verlag. 162, 166
- Portinale, L.; Torasso, P.; and Tavano, P. 1998. Proceedings of the thirteenth european conference on artificial intelligence. In *Dynamic Case Memory Management*, 73–77. Chichester: Wiley.
- Portinale, L.; Torasso, P.; and Tavano, P. 1999. Speed-up, quality, and competence in multi-modal reasoning. In *Proceedings of the Third International Conference on Case-Based Reasoning*, 303–317. Berlin: Springer Verlag. 161, 162, 163, 170
- Smyth, B., and Cunningham, P. 1996. The utility problem analysed: A case-based reasoning perspective. In *Proceedings of the Third European Workshop on Case-Based Reasoning*, 392–399. Berlin: Springer Verlag. 162
- Smyth, B., and Keane, M. 1995. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 377–382. Montreal: IJCAI. 161
- Smyth, B., and Keane, M. 1998. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence* 102(2):249–293. 171
- Smyth, B., and McKenna, E. 1999a. Building compact competent case-bases. In *Proceedings of the Third International Conference on Case-Based Reasoning*. Berlin: Springer Verlag. 161, 163, 166, 169, 171
- Smyth, B., and McKenna, E. 1999b. Footprint-based retrieval. In *Proceedings of the Third International Conference on Case-Based Reasoning*. Berlin: Springer Verlag.
- van Someren, M.; Surma, J.; and Torasso, P. 1997. A utility-based approach to learning in a mixed case-based and model-based reasoning architecture. In *Proceedings of the Second International Conference on Case-Based Reasoning*, 477–488. Berlin: Springer Verlag.
- Zhu, J., and Yang, Q. 1999. Remembering to add: Competence-preserving case-addition policies for case base maintenance. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann. 161, 163, 165

# Case-Based Reasoning for Breast Cancer Treatment Decision Helping

Jean Lieber and Benoît Bresson

Orpailleur research group  
LORIA, UMR 7503 – CNRS, INRIA-Lorraine, Nancy Universities  
BP 239, 54506 Vandœuvre-lès-Nancy, France  
{lieber,bresson}@loria.fr

**Abstract.** This paper presents two applications for the breast cancer treatment decision helping. The first one is called CASIMIR/RBR and can be likened to a rule-based reasoning system. In some situations, the application of the rules of this system does not provide a satisfying treatment. Then, the application CASIMIR/CBR—which is not fully implemented—can be used. CASIMIR/CBR uses principles of case-based reasoning in order to suggest solutions by adapting the rules of CASIMIR/RBR. In this framework, the rules are considered as cases: they are adapted rather than used literally.

**Keywords:** decision helping, breast cancer treatment, rules as cases.

## 1 Introduction

Case-based reasoning (CBR, [10,1]) is a kind of reasoning which aims at solving a problem, called the *target* problem, thanks to a similar *case*. A case is given by a problem and a solution to this problem. The *case base* is the set of the available cases. The first step of CBR, *retrieval*, consists in choosing a case in the case base according to similarity criteria. During the *adaptation* step, the retrieved case is adapted in the perspective of the target problem solving. Finally, the *learning* step consists in integrating, if necessary, the new case—the target problem and its solution, once this latter is validated—in the case base.

CBR has been often contrasted with rule-based reasoning (RBR). This contrast can be seen according to two different viewpoints. First, the cases are generally *specific* knowledge linked with experiments in the “real world”, whereas rules are *general* knowledge linked with “theories” about the world. The second viewpoint, more important in the framework of this paper, is about the way the knowledge units—rules or cases—are used:

- The rules are (typically) used *literally*; the conclusion of the rule is simply instantiated.
- The cases are *adapted* and not simply copied or instantiated.

According to this second viewpoint, the difference between a case and a rule is not based on their contents but on the different ways they are used (literally for RBR and by adaptation for CBR).

Thus, the same knowledge units can be used to do reasoning from rules or from cases. In particular, a rule  $R = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  can be seen as a case whose problem is the premise **Prem** of **R** and whose solution is the conclusion **Ccl**<sup>°</sup> of **R**. This is the principle that has been adopted for the two applications CASIMIR/RBR –which can be seen as a RBR application– and CASIMIR/CBR –which can be seen as a CBR application. Indeed, these applications are based on the same knowledge base whose domain is the breast cancer treatment. Note that the first application is now fully implemented whereas the second one is not.

Section 2 describes the way decisions are taken in breast cancer treatment. CASIMIR/RBR is briefly presented in section 3. CASIMIR/CBR is presented with more details in section 4. Section 5 briefly describes the approach for the adaptation knowledge acquisition and modelisation. Section 6 explains how, thanks to these applications, the knowledge base can be updated. Finally, section 7 compares these applications with RESYN and RESYN/CBR, two applications dedicated to synthesis planning in organic chemistry. Section 8 concludes the paper and presents some future work.

## 2 Decision Making for Breast Cancer Treatment

This work is part of a project involving specialists of cancer from the *Centre Alexis Vautrin* (CAV, Vandœuvre-lès-Nancy), ergonomics specialists from the CNAM and computer scientists from the Orpailleur research group of the LORIA (see the acknowledgements at the end of the paper for more informations about the people involved in this project). Its goal is to specify and represent in a computer system the decision making process for treating breast cancers. This section briefly describes how cancer specialists from the CAV do the reasoning leading to a decision. The reasoning is based on the so-called *protocol*, which represents the knowledge for decision making and which is based on statistical studies. The protocol can thus be seen as a set of rules for the use of cancer specialists. In some cases, the protocol cannot be used, at least in two types of situations:

- (S1) it is not applicable on the current patient (such a situation occurs because the rule base is not complete).
- (S2) it is applicable, but the treatment suggested is problematical (treatment impossible to apply or leading to unexpected difficulties).

In each of the situation types (S1) and (S2), the regular doctor of the patient brings the case to the *breast therapeutic decision committee* (called “breast TDC” in the following). The breast TDC is a multidisciplinary advisory board made of cancer specialists from the CAV (specialists of chemotherapy, surgery, hormoneotherapy and radiotherapy) who are meeting themselves at regular intervals. The doctor formulates a question to which the TDC has to suggest an answer. The reasoning leading to this answer is not independent from the protocol nor, of course, does it come from a direct application of it (since, in this situation,

the protocol answer is not satisfying). This reasoning is an adaptation of the protocol to the current situation. Finally, the doctor takes a decision based on the propositions of the breast TDC.

The decision making can thus be divided into one or two steps: the first step is a protocol-based reasoning implemented in the CASIMIR/RBR system thanks to rule-based reasoning and the second one (which is not systematic, since it involves only 30% of the patients) is implemented in CASIMIR/CBR thanks to case-based reasoning.

### 3 CASIMIR/RBR

The main difficulty in the conception of CASIMIR/RBR was the constitution of the knowledge base. It can be seen as a translation of the knowledge represented in the protocol. The difficulties, in this translation, mainly come from the important part of the implicit knowledge contained in the protocol. Thus, it was mainly a work of making the knowledge explicit.

In the following sections, it is considered that the protocol is represented by a set of rules and that the reasoning done by CASIMIR/RBR is a RBR.  $\text{pat}^{\text{nt}}$  denotes the computer representation of a patient and of their tumour. The rule  $R = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  can be applied to  $\text{pat}^{\text{nt}}$ , when the set of the characteristics of  $\text{pat}^{\text{nt}}$  entails Prem, which is denoted by

$$\text{Prem} \Leftarrow \text{pat}^{\text{nt}} \quad (1)$$

For example, let R be the following rule:

$$R = \boxed{\begin{array}{c} S \in [0; 4] \\ N = + \\ M = - \\ MFT = \text{yes} \\ \text{sex} = \text{female} \\ \hline \text{Patey} \end{array}} \quad (2)$$

The premise of R is:

$$\text{Prem} = (S \in [0; 4]) \wedge (N = +) \wedge (M = -) \wedge (MFT = \text{yes}) \wedge (\text{sex} = \text{female})$$

$S$  is the size of the tumour (in centimetres).  $N$  indicates whether there are cancerous ganglions ( $N = +$ ) or not ( $N = -$ ).  $M$  indicates whether there is a metastasis or not ( $M = +$  or  $M = -$ ).  $MFT$  indicates whether a tumour is multifocal or not ( $MFT = \text{yes}$  or  $MFT = \text{no}$ ): a tumour is said to be multifocal when it appears as a set of disconnected tumours. Last, **sex** indicates the sex of the patient (**male** or **female**).

The conclusion of R is:

$$\text{Ccl}^\circ = \text{Patey}$$

A Patey is a surgical operation which consists in a breast ablation.

Let  $\text{pat}^{\text{nt}}$  be the following representation of a patient (and their tumour):

$$\text{pat}^{\text{nt}} = \begin{array}{l} S = 2.5 \\ N = + \\ M = - \\ MFT = \text{yes} \\ \text{sex} = \text{female} \\ \text{age} = 55 \end{array}$$

Since  $\text{Prem} \Leftarrow \text{pat}^{\text{nt}}$ , it can be concluded that the rule  $R$  can be applied on  $\text{pat}^{\text{nt}}$ . Hence, the protocol recommends a Patey for the patient represented by  $\text{pat}^{\text{nt}}$ .

From a practical point of view, the rule base is implemented as a hierarchy of classes, and the reasoning is a classification reasoning in this hierarchy. About 30 attributes are necessary to represent the rule premises and the patients and there are several hundred classes (representing rules).

CASIMIR/RBR can be likened to the application ONCODOC that implements another protocol for the breast cancer treatment [11]. Besides the difference between these protocols, these two applications are different in particular in the knowledge implementation formalisms: the ONCODOC protocol is represented by a set of interrelated hypertext documents whereas the CASIMIR/RBR protocol is represented by a hierarchy of classes representing rules. This latter formalism has seemed to us more adapted to the non literal use of the protocol, such as the one carried out by CASIMIR/CBR.

## 4 CASIMIR/CBR

This section presents the way the development of the application CASIMIR/CBR is scheduled.

The reasoning of the breast TDC that CASIMIR/CBR will have to simulate is composed of three steps:

**Elaboration:** the question associated with  $\text{pat}^{\text{nt}}$  is highlighted in order to make clear what part of  $\text{pat}^{\text{nt}}$  actually raises a problem with respect to the protocol (for elaboration in CBR, see [5]).

**Retrieval:** the rule  $R = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  *closest* to  $\text{pat}^{\text{nt}}$  is searched in the rule base.

**Adaptation:**  $\text{Ccl}^\circ$ , the treatment associated to this rule, is adapted in order to suggest a treatment for the patient.

In section 2, situations types (S1) and (S2) have been described. The sections 4.1 and 4.2 describe CASIMIR/CBR in situations of types (S1) and (S2).

#### 4.1 Situations of Type (S1): When Casimir/RBR Does not Return any Treatment

A patient is in a (S1) situation when the rules cannot be used in order to solve the target problem. This means that, for whatever rule  $R = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  in the rule base, the relation (1) is not satisfied. The idea is then to try to obtain this relation approximatively, i.e. with some authorised modifications of  $\text{Prem}$  and  $\text{pat}^{\text{nt}}$ . Let  $\varphi(\text{Prem})$  be the modified premise of the rule and  $\psi(\text{pat}^{\text{nt}})$  be the modified patient. If the modifications are appropriate, then the relation (1) holds when  $\text{Prem}$  is substituted by  $\varphi(\text{Prem})$  and  $\text{pat}^{\text{nt}}$  is substituted by  $\psi(\text{pat}^{\text{nt}})$ ; written in another way:

$$\text{Prem} \xrightarrow{\varphi} \varphi(\text{Prem}) \Leftarrow \psi(\text{pat}^{\text{nt}}) \xleftarrow{\psi} \text{pat}^{\text{nt}} \quad (3)$$

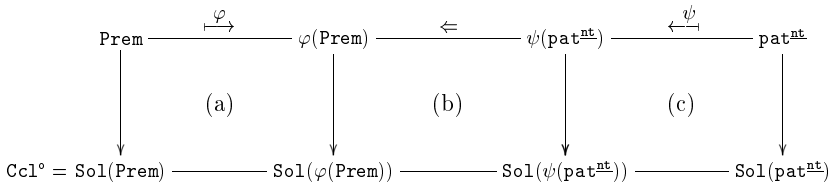
Such a sequence of relations is called a *similarity path*. Such a path is composed of *relations* ( $\xrightarrow{\varphi}$ ,  $\Leftarrow$  and  $\xleftarrow{\psi}$ ) and of *problems*. Among the problems, there are the “general” problems  $\text{Prem}$  and  $\varphi(\text{Prem})$  –at the level of generality of the rules– and “specific” problems  $\psi(\text{pat}^{\text{nt}})$  and  $\text{pat}^{\text{nt}}$  –at the level of generality of the representation of patients. These problems come from different origins:

- $\text{pat}^{\text{nt}}$  is the input of the application.
- $\text{Prem}$  is a problem *to be chosen* among the premises of the rules of the base.
- $\varphi(\text{Prem})$  and  $\psi(\text{pat}^{\text{nt}})$  are problems *built* in order to relate  $\text{Prem}$  and  $\text{pat}^{\text{nt}}$  thanks to a similarity path.

These two problems are called *intermediate problems*: they are built to be used as intermediaries between  $\text{Prem}$  and  $\text{pat}^{\text{nt}}$ .

Although retrieval comes before adaptation, this latter is described first in the following. Indeed, retrieval aims at providing a case to be adapted: its specification is therefore linked to the adaptation process.

**Adaptation.** The adaptation of the conclusion  $\text{Ccl}^\circ$  of the rule  $R = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  is based on the similarity path (3). In the following,  $\text{Ccl}^\circ$  is denoted by  $\text{Sol}(\text{Prem})$ : in terms of problems and solutions,  $\text{Ccl}^\circ$  is the solution associated with  $\text{Prem}$ . The adaptation takes three steps, one for each relation of the similarity path. This process can be summarised by the following scheme:



- (a) adaptation of  $\text{Ccl}^\circ = \text{Sol}(\text{Prem})$  in a solution  $\text{Sol}(\varphi(\text{Prem}))$  of  $\varphi(\text{Prem})$ ;
- (b) adaptation of  $\text{Sol}(\varphi(\text{Prem}))$  in a solution  $\text{Sol}(\psi(\text{pat}^{\text{nt}}))$  of  $\psi(\text{pat}^{\text{nt}})$ ;
- (c) adaptation of  $\text{Sol}(\psi(\text{pat}^{\text{nt}}))$  in a solution  $\text{Sol}(\text{pat}^{\text{nt}})$  of  $\text{pat}^{\text{nt}}$ .



The adaptation step (b) is very simple: since a solution  $\text{Sol}(\varphi(\text{Prem}))$  of the problem  $\varphi(\text{Prem})$  is known, an *adapted rule*  $\text{AR} = (\varphi(\text{Prem}) \longrightarrow \text{Sol}(\varphi(\text{Prem})))$  can be used. In particular,  $\text{AR}$  can be used to solve  $\psi(\text{pat}^{\text{nt}})$  since  $\varphi(\text{Prem}) \Leftarrow \psi(\text{pat}^{\text{nt}})$ .

Adaptation steps (a) and (c) are linked with the available adaptation knowledge. This knowledge is assumed to be represented by ordered pairs  $(\mathbf{r}, \mathcal{A}_{\mathbf{r}})$ , called *reformulations* [9].  $\mathbf{r}$  is a binary relation between problems and  $\mathcal{A}_{\mathbf{r}}$  is a mapping.  $\mathcal{A}_{\mathbf{r}}$  takes as inputs a source problem  $\text{srce}$ , a solution  $\text{Sol}(\text{srce})$  of  $\text{srce}$  and a target problem  $\text{tgt}$  such that  $\text{srce} \mathbf{r} \text{tgt}$ . The output of  $\mathcal{A}_{\mathbf{r}}$  is a solution  $\text{Sol}(\text{tgt})$  of  $\text{tgt}$ :

$$\mathcal{A}_{\mathbf{r}} : (\text{srce}, \text{Sol}(\text{srce}), \text{tgt}) \mapsto \text{Sol}(\text{tgt})$$

The adaptation steps (a) and (c) are therefore possible if there are adaptation knowledge given by appropriate reformulations. That is why the choice of  $\varphi(\text{Prem})$  and of  $\psi(\text{tgt})$  must be done in accordance with these reformulations.

It must be noticed that each of the steps (a) and (b) can itself be decomposed in substeps. For instance, let  $\left( \overset{\varphi_1}{\underset{\varphi_1}{\longrightarrow}} \mathcal{A}_{\varphi_1} \right)$  and  $\left( \overset{\varphi_2}{\underset{\varphi_2}{\longrightarrow}} \mathcal{A}_{\varphi_2} \right)$  be two reformulations and let  $\varphi = \varphi_2 \circ \varphi_1$ . Then, the solution associated with  $\text{Prem}$  can be adapted in a solution associated with  $\varphi(\text{Prem})$ :

- (a.i)  $\text{Sol}(\text{Prem})$  is adapted into a solution  $\text{Sol}(\varphi_1(\text{Prem}))$  of  $\varphi_1(\text{Prem})$ ;
- (a.ii)  $\text{Sol}(\varphi_1(\text{Prem}))$  is adapted into a solution  $\text{Sol}(\varphi(\text{Prem}))$  of  $\varphi_2(\varphi_1(\text{Prem})) = \varphi(\text{Prem})$ .

In a similar way, (c) can be decomposed in several substeps. Thus, the number of reformulations used in a similarity path is not a priori limited.

The knowledge acquisition and modelisation that leads to the available reformulations is briefly discussed in section 5. Two examples of reformulations are detailed below.

*Example 1.* Let  $\text{pat}^{\text{nt}}$  be the above description of a patient:

$$\text{pat}^{\text{nt}} = \begin{array}{c} S = 4.2 \\ N = + \\ M = - \\ MFT = \text{yes} \\ \text{sex} = \text{female} \\ \text{age} = 38 \end{array}$$

It is assumed that the rule to be adapted to  $\text{pat}^{\text{nt}}$  is the rule  $\text{R}$  given by equation (2). This rule cannot be used in a literal way:  $\text{Prem} \not\Leftarrow \text{pat}^{\text{nt}}$ . Nevertheless, it can be seen that there is not much modification to do in order to make this rule applicable: with an increasing of 2 millimetres of the maximum tumour size

associated with the rule, the rule becomes applicable. So, let **AR** be the following rule (adapted from **R**):

$$\mathbf{AR} = \boxed{\begin{array}{c} S \in [0; 4.2] \\ N = + \\ M = - \\ MFT = \text{yes} \\ \hline \text{sex} = \text{female} \\ \text{Patey} \end{array}}$$

This rule  $\mathbf{AR} = (\varphi(\text{Prem}) \longrightarrow \text{Sol}(\varphi(\text{Prem})))$  can be applied on  $\text{pat}^{\text{nt}}$ :  $\varphi(\text{Prem}) \leftarrow \text{pat}^{\text{nt}}$ . Therefore, the treatment associated with **AR** can be applied to  $\text{pat}^{\text{nt}}$ .

The adaptation performed consists in weakening the premise of the rule to be adapted and to copy its solution:  $\text{Sol}(\varphi(\text{Prem})) := \text{Sol}(\text{Prem})$ .

Thus there is no modification of the solution in this adaptation, but the confidence on the adapted rule **AR** is smaller than the confidence associated with **R**. This “confidence decreasing” can be measured by the weakening done from **Prem** to  $\varphi(\text{Prem})$ : the more the difference between the maximum size of the tumour is, the less reliable the result is.

For this reformulation and similar reformulations, an implementation based on fuzzy attributes can be envisaged.

*Example 2.* Since the rules of the base are obtained from statistical studies and since the breast cancer is rare for the men, every rule of the base contains the equality  $\text{sex} = \text{female}$  in its premise. Therefore, a male patient will not have a treatment proposed by the literal application of the rules. Faced to such a problem, the breast TDC have to propose a treatment. The idea is then to adapt the treatment that would have been proposed if the patient were a woman, every other attributes being the same.

In order to model this kind of adaptation, the reformulation  $\left( \overset{\psi}{\leftarrow} \mathcal{A} \overset{\psi}{\leftarrow} \right)$  can be proposed, with

- $\psi(\text{pat}^{\text{nt}})$  being the patient transformed into a woman (change of the attribute  $\text{sex}$ , and of the sex-dependent attributes);
- If  $\psi(\text{pat}^{\text{nt}})$  has a solution  $\text{Sol}(\psi(\text{pat}^{\text{nt}}))$ , this treatment is adapted into a solution  $\text{Sol}(\text{pat}^{\text{nt}})$  of  $\text{pat}^{\text{nt}}$ .

For instance, if  $\text{Sol}(\psi(\text{pat}^{\text{nt}}))$  contains “ablation of the ovaries”,  $\text{Sol}(\text{pat}^{\text{nt}})$  will contain “ablation of the testicles”.

**Retrieval.** Retrieval consists in searching a rule  $\mathbf{R} = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  in the base and to build a similarity path between **Prem** and  $\text{pat}^{\text{nt}}$  given by the modifications  $\varphi(\text{Prem})$  of **Prem** and  $\psi(\text{pat}^{\text{nt}})$  of  $\text{pat}^{\text{nt}}$  such that  $\varphi(\text{Prem}) \leftarrow \psi(\text{pat}^{\text{nt}})$ . As discussed above,  $\varphi$  and  $\psi$  are compositions of mappings  $\varphi_i$ 's and  $\psi_j$ 's such that

$\left(\overset{\varphi^i}{\vdash} \mathcal{A}_{\varphi^i}\right)$  and  $\left(\overset{\psi^j}{\vdash} \mathcal{A}_{\psi^j}\right)$  are available reformulations. Thus, a similarity path, detailed at the level of reformulations, has the following form:

$$\text{sim-path} = \left( \text{Prem} \overset{\varphi^1}{\vdash} \dots \overset{\varphi^p}{\vdash} \varphi(\text{Prem}) \Leftarrow \psi(\text{pat}^{\text{nt}}) \overset{\psi^q}{\vdash} \dots \overset{\psi^1}{\vdash} \text{pat}^{\text{nt}} \right)$$

with  $\varphi = \varphi^p \circ \dots \circ \varphi^1$  and  $\psi = \psi^q \circ \dots \circ \psi^1$ .

Each of the  $(p + 1 + q)$  adaptation steps may decrease the confidence on the produced solution. If a numerical value  $\text{cost}(\mathbf{r}) \geq 0$ , characteristic of this confidence decreasing, is associated with each of the relations  $\mathbf{r}$  of the similarity path  $\text{sim-path}$  ( $\mathbf{r} = \overset{\varphi^i}{\vdash} \mathbf{r} = \Leftarrow$  or  $\mathbf{r} = \overset{\psi^j}{\vdash}$ ), a cost can be associated to  $\text{sim-path}$ :

$$\text{cost}(\text{sim-path}) = \sum_{\mathbf{r}, \text{ relation of sim-path}} \text{cost}(\mathbf{r})$$

Minimising this cost remains to maximise the confidence on the solution produced by adaptation. There is sometimes several similarity paths relating a given premise of rule **Prem** to  $\text{pat}^{\text{nt}}$ . The path that will have to be chosen is the one minimising this cost. Let **dist** be defined as follows:

$$\text{dist}(\text{Prem}, \text{pat}^{\text{nt}}) = \min_{\text{sim-path}} \text{cost}(\text{sim-path})$$

the minimum being taken on the set of the similarity paths relating **Prem** to  $\text{pat}^{\text{nt}}$ . The *best* rule  $\mathbf{R} = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  for adaptation to patient is the one minimising  $\text{dist}(\text{Prem}, \text{pat}^{\text{nt}})$  and that is it that the retrieval process must return.

A similarity path between **Prem** and  $\text{pat}^{\text{nt}}$  can be found thanks to a search in a state space such that (1) a state is an ordered pair of problems  $(\mathbf{pb}, \mathbf{pb}')$ , (2) the initial state is  $(\text{Prem}, \text{pat}^{\text{nt}})$ , (3) the successors of  $(\mathbf{pb}, \mathbf{pb}')$  are the states  $(\varphi(\mathbf{pb}), \mathbf{pb}')$  and the states  $(\mathbf{pb}, \psi(\mathbf{pb}'))$  where  $\left(\overset{\varphi}{\vdash} \mathcal{A}_{\varphi}\right)$  and  $\left(\overset{\psi}{\vdash} \mathcal{A}_{\psi}\right)$  are available reformulations, (4) the final state is such that  $\mathbf{pb} \Leftarrow \mathbf{pb}'$ . If  $(\mathbf{pb}, \mathbf{pb}')$  is a final state, then  $\text{Prem} \vdash \dots \vdash \mathbf{pb} \Leftarrow \mathbf{pb}' \vdash \dots \vdash \text{pat}^{\text{nt}}$  is a similarity path. The smooth classification is a retrieval technique combining this search with hierarchical classification that provides a similarity path together with the retrieved case (more details are presented in [7] and [8]).

It must be noticed that retrieval does not only provide a similar case but also a similarity path that is used as a guideline for adaptation. Thus, the retrieved rule is necessarily adaptable: the retrieval process follows the adaptation-guided retrieval philosophy [12]. Moreover, the retrieval may return several adaptable cases with several similarity paths. Therefore, several different treatments may be returned by the adaptation. **dist** is used to present these treatments in an ordered way (starting with the one adapted from the rule  $\mathbf{R} = (\text{Prem} \longrightarrow \text{Ccl}^\circ)$  minimising  $\text{dist}(\text{Prem}, \text{pat}^{\text{nt}})$ ). The treatments returned by CASIMIR/CBR must be considered as argued suggestions that may help the work of the breast TDC.

**Elaboration.** Before going to the breast TDC, the regular doctor of the patient fills a form describing the patient and their tumour, as well as the question he/she

wants to ask to the breast TDC. Since the rule base does not provide a treatment, the question could be very general: “What is the treatment?”. But to make the work of the breast TDC more efficient, the regular doctor is led to formulate its question more accurately. CASIMIR/RBR is supposed to help the question formulation. This step is called *elaboration* of the target problem: it highlights the problem to be solved. Elaboration can follow immediately retrieval. If the rule R is the closest rule to  $\text{pat}^{\text{nt}}$  according to  $\text{dist}$ , CASIMIR/RBR may propose the following question: “Is it possible to modify the rule R in order to treat the patient and how? ”.

For situations of type (S1), elaboration appears to be a “subproduct” of retrieval. This is not true for situations of type (S2).

## 4.2 Situations of Type (S2): When Casimir/RBR Returns an Inadequate Treatment

A patient is in a situation of type (S2) when CASIMIR/RBR proposes a treatment whose application raises some difficulties.

For example, the chemotherapy proposed can be incompatible with heart problems. Then, the proposed treatment must be modified. In these circumstances, the contraindicated drug can be substituted by another drug with a similar desired effect but that does not raise a contraindication problem.

Let us consider this other example: the treatment consists in an ablation of the ovaries of a woman who does not have any child and wants to. In this case, the doctor would have scruples to prescribe this surgery act. The breast TDC has to decide whether this act will have to be done and, if not, has to indicate a substitutive treatment.

The elaboration step is very important in the (S2) situations. Indeed, it indicates what should be modified in the treatment proposed by CASIMIR/RBR. For the moment, it is scheduled that the question will be given by the user. A long term future work would be to implement a question formulation help module.

The retrieval step is very simple in (S2) situations: the rule used by CASIMIR/RBR is the one which will have to be adapted and, thus, returned by retrieval.

The adaptation step in (S2) situations remains to be studied deeply. We plan to implement it in the following way. First, the analysis of the question must explain which parts of the treatment raise some difficulties and may have to be adapted. Second, the other part of the treatment is simply copied: if the only difficulty is about the chemotherapy, it is useless to modify the radiotherapy. Third, the adaptation process aims at substituting the treatments that raise difficulties by treatments with similar desired effects but which does not raise these difficulties.

## 5 Adaptation Knowledge Acquisition and Modelisation

The adaptation knowledge is represented by a set of reformulations  $(\mathbf{r}, \mathcal{A}_{\mathbf{r}})$ , as noted above for the (S1) situations (it is assumed here that it is also the case

for the (S2) situations). This section describes the approach that emerged for the acquisition of reformulations. An attempt of generalisation of this approach should be a part of the future work.

The persons involved in the acquisition of reformulations meetings are specialists of cancer, of ergonomics and of computer science. Each of these meetings consists in the study of an adaptation that was performed during a breast TDC session. This study produces a decomposition of the adaptation in several simple steps which are translated into a similarity path **sim-path**. **sim-path** relates the premise of a rule from the rule base to the description of a patient, and contains intermediate problems representing “virtual patients”. Then, each of the relations of **sim-path** is analysed in association with the corresponding adaptation step. This analysis leads to a proposed reformulation which is discussed with the experts.

Although this approach is rather simple and should be improved, it has turned out to be very fruitful.

## 6 Evolutions of the Knowledge Base

The knowledge used in CASIMIR/RBR is represented by a rule base. The knowledge used in CASIMIR/CBR is represented by the rule base –used as a case base– and by a set of reformulations constituting the adaptation knowledge. This latter can be seen as a set of *protocol correcting rules*. Sometimes, these knowledge bases are not sufficient to provide a treatment, or may lead to an inadequate treatment, considering the current knowledge about breast cancer (which is in constant evolution). In such a situation, the reasoning leads to a failure. In order to revise and update the knowledge bases, we plan to study and implement a mechanism of *failure-based learning*. This learning mechanism consists in analysing the failure and attempting to build, with the help of breast cancer specialists, new rules for which the failure does not occur any more in the same situation (or in similar situations) [3]. This analysis is made in cooperation with the experts.

The failure-based learning enables to progressively improve the adaptation knowledge. When this latter is considered to be reasonably stable, it can be integrated in the rule base. Thus, this learning process involves an improvement of the protocol. This kind of learning can be likened to the research on explaining and repairing failures for the case-based planner CHEF [6].

## 7 Two Similar Works: Resyn and Resyn/CBR

The use of RBR (in fact, of hierarchical classification) in order to model the reasoning of CASIMIR/RBR is deeply inspired from the system RESYN [13] whose domain is the synthesis planning in organic chemistry. RESYN has been implemented in the framework of a CNRS research group about computer treatment of knowledge in organic chemistry.

Another work done in the framework of this research group was the use of CBR for the same application domain. The system RESYN/CBR [7] can be seen as an extension of RESYN, the difference being essentially due to the fact that RBR is followed by CBR in RESYN/CBR. In particular, the notions of similarity path and of intermediate problem have been highlighted during this work on RESYN/CBR.

The part of the application CASIMIR/CBR which should be the most similar to RESYN/CBR is the one that deals with (S1) situations. Indeed, a similarity path analogous to the one described by equation (3) is built during the RESYN/CBR retrieval and is used during its adaptation. The technique used for retrieval in RESYN/CBR is smooth classification. For the part of CASIMIR/CBR dealing with (S1) situations, the same technique is scheduled.

## 8 Conclusion and Future Work

This paper presents two applications, CASIMIR/RBR and CASIMIR/CBR, that are dedicated to decision helping in the framework of breast cancer treatment. The first one is an application performing a “classical” reasoning based on rules. The second one uses case-based reasoning in an unusual way: the rules of CASIMIR/RBR are used as cases for CASIMIR/CBR. This approach is justified by the idea that this is how you use it that makes a knowledge unit a rule or a case. These rules considered as cases can be likened to the *ossified cases* defined in [10].

If CASIMIR/RBR is fully implemented, this is not the case for CASIMIR/CBR. This paper presents the general principles that we plan to follow, principles that, for an important part, come from the realisation of RESYN/CBR, a case-based reasoning application dedicated to synthesis planning in organic chemistry.

The future work is essentially the realisation of CASIMIR/CBR. The first issue is the adaptation knowledge acquisition by the study of the breast TDC sessions. The second issue is the representation of solutions. Indeed, to perform adaptations, it is useful to be able to manipulate solutions at different representation levels. The researches on abstraction in CBR (see e.g. [2] and [4]) should be useful for this issue. The third issue is the implementation of elaboration, retrieval and adaptation steps when either CASIMIR/RBR cannot solve the target problem or solve it inadequately. Elaboration should be useful for helping the question formulation preparing the breast TDC sessions. Retrieval and adaptation are supposed to help the decision during these sessions. In general, many issues presented in this paper must be made more precise. For example, the study on failure-based learning must go on.

A long term future work consists in integrating in the case base “real cases” of patients together with their treatment and not only cases used as rules by CASIMIR/RBR. This study would enable to see how the use of specific cases differs from the use of general cases (differences in the adaptation knowledge, etc.). It would also be interesting to see how the specific cases could be used for

elaboration, in particular in the situations for which CASIMIR/RBR provides inadequate treatments.

## Acknowledgements

The authors would like to thank the anonymous referees whose suggestions and comments greatly contributed to improve the preliminary versions of this paper. It was not possible to discuss in this paper all the issues they have highlighted, but this will be helpful for future work.

This work takes place in an interdisciplinary context that involves cancer specialists from the CAV, ergonomics specialists from the CNAM as well as computer scientists from the Orpailleur research group of the LORIA. We would like to thank here all the people involved in this project and more particularly, in alphabetical order: Professor Pierre Bey, Pierre Falzon, Doctor Anne Lesur, Amedeo Napoli, Doctor Maria Rios and Doctor Catherine Sauvagnac. We would also like to thank Grégoire Boussu whose work in the context of this project during his “DEA” in computer science (French equivalent of the master degree) is useful for us. Finally, we would like to thank Mathieu Daquin for the development of the hierarchy visualisation tool and Jean-Yves Burlett for implementing various functionalities for CASIMIR/RBR (user interface, file management, etc.).

## References

1. A. Aamodt and E. Plaza. Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994. 173
2. R. Bergmann and W. Wilke. Building and Refining Abstract Planning Cases by Change of Representation Language. *Journal of Artificial Intelligence Research*, 3:53–118, 1995. 183
3. G. Boussu. Apprentissage à partir d'échecs. Mémoire de DEA d'informatique, Université Henri Poincaré Nancy 1, 1998. 182
4. L. K. Branting and D. W. Aha. Stratified Case-Based Reasoning: Reusing Hierarchical Problem Solving Episodes. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montréal, volume 1, pages 384–390, August 1995. 183
5. Béatrice Fuchs, Jean Lieber, Alain Mille, and Amedeo Napoli. Towards a Unified Theory of Adaptation in Case-Based Reasoning. In K.-D. Althoff, R. Bergmann, and L. K. Branting, editors, *Case-Based Reasoning Research and Development — Third International Conference on Case-Based Reasoning (ICCBR-99)*, Lecture Notes in Artificial Intelligence 1650. Springer, Berlin, 1999. 176
6. K. J. Hammond. Explaining and Repairing Plans That Fail. *Artificial Intelligence*, 45:173–228, 1990. 182
7. J. Lieber and A. Napoli. Using Classification in Case-Based Planning. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96)*, Budapest, Hungary, pages 132–136. John Wiley & Sons, Ltd., 1996. 180, 183

8. J. Lieber and A. Napoli. Correct and Complete Retrieval for Case-Based Problem-Solving. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, Brighton, United Kingdom, pages 68–72, 1998. 180
9. E. Melis, J. Lieber, and A. Napoli. Reformulation in Case-Based Reasoning. In B. Smyth and P. Cunningham, editors, *Fourth European Workshop on Case-Based Reasoning, EWCBR-98*, Lecture Notes in Artificial Intelligence 1488, pages 172–183. Springer, 1998. 178
10. C. K. Riesbeck and R. C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989. 173, 183
11. B. Séroussi, J. Bouaud, and É.-C. Antoine. Enhancing Clinical Practice Guideline Compliance by Involving Physicians in the Decision Process. In W. Horn, Y. Shahr, G. Lindberg, S. Andreassen, and J. Wyatt, editors, *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM-99)*, volume 1620 of *LNAI*, pages 76–85, Berlin, 1999. Springer. 176
12. B. Smyth and M. T. Keane. Using adaptation knowledge to retrieve and adapt design cases. *Knowledge-Based Systems*, 9(2):127–135, 1996. 180
13. P. Vismara. Reconnaissance et représentation d'éléments structuraux pour la description d'objets complexes. Application à l'élaboration de stratégies de synthèse en chimie organique. Thèse de l'Université des Sciences et Techniques du Languedoc, Montpellier, 1995. 182



# Competence-Guided Case-Base Editing Techniques

Elizabeth McKenna and Barry Smyth

Smart Media Institute, Department of Computer Science  
University College Dublin, Belfield, Dublin 4, Ireland  
{Elizabeth.McKenna, Barry.Smyth}@ucd.ie

**Abstract.** Case-based classification is a powerful classification method, which (in its simplest form) assigns a target case to the same class as the nearest of  $n$  previously classified cases. Many case-based classifiers use the simple nearest-neighbour technique to identify the nearest case, but this means comparing the target case to all of the stored cases at classification time, resulting in high classification costs. For this reason many techniques have been proposed to improve the performance of case-based classifiers by reducing the search they must perform. In this paper we will look at editing techniques that preserve the lazy-learning quality of case-based classification, but improve classification performance.

## 1 Introduction

Case-based reasoning (CBR) can be used to implement simple but powerful classification systems. The class of an unclassified target case is determined by examining the class of the  $k$  nearest cases in the case-base. Many case-based classifiers use a simple nearest-neighbour (NN) retrieval approach whereby the target case is compared to each and every case in the case-base to determine the  $k$  nearest neighbours (kNN). This approach has two basic problems. First, the classification cost can be high because it is expensive to compare a target case to a large number of stored cases. Second, noise in the case-base can reduce overall classification accuracy.

The problem of a high classification cost is often addressed by pre-processing the case-base in order to generate a representation that facilitates a more efficient search. For example,  $k$ -d trees constructed over the case-base as part of the learning process can reduce the classification cost to  $O(\log_2 n)$  in the number of cases. Another alternative is to *edit* the case-base by removing redundant or harmful cases. There are a number of advantages to this approach. First, the classification cost problem is addressed because the edited case-base is typically much smaller than the original case-base. Second, classification accuracy can be improved by identifying and removing noisy cases as part of the editing process. Editing techniques have a long history in the pattern-recognition community [1,5,6,7,8,9,10,17,18,19]. Recently these techniques have been advanced by the case-based learning community through the work of Aha et al. [1], among others.

Many of the editing techniques used by these communities translate effectively as case-base editing techniques. In this paper we will describe and evaluate a number of new case-based editing methods that have been inspired by traditional editing algorithms and recent work by the CBR community on competence modelling [12,13,14]. In short, we will show how different local and global competence metrics can be used to effectively guide the case editing process, and how the resulting edited case-bases benefit from superior classification performance under a wide range of experimental conditions. The work reported here is a significant extension of our previous work [14], both in terms of the techniques described and the evaluation carried out. Moreover, it represents a more detailed and focused analysis of a group of three editing techniques that form part of a larger family of methods described and evaluated elsewhere [11].

## 2 Related Work

There are two basic reasons for editing training data - redundancy reduction and noise removal. Redundancy reduction algorithms aim to eliminate training cases that do not contribute to classification competence, for example, cases from the interior of a densely packed class cluster. These algorithms can produce significant training set reductions but tend to preserve noisy examples as apparent exceptions. In contrast, noise reduction techniques aim to eliminate noisy examples from the training set. These algorithms tend to result in less dramatic size reductions (depending on the level of training set noise) and they can also mistakenly remove important exceptional cases, which are difficult to distinguish from true noise. In the remainder of this section we will survey a representative sample of past and present case selection techniques, many coming from the nearest-neighbour classification and instance-based learning communities - for reasons of clarity we will describe these techniques in the context of a case-base classifier, and thus talk about cases rather than instances, for example. We do this without loss of generality.

### 2.1 The CNN Family

Hart's condensed nearest neighbour (CNN) approach is probably the earliest and best-known editing method [9]. It is a redundancy reduction technique that incrementally builds an edited case-base from scratch. Cases are added to this set, and removed from the original case-base, if and only if it cannot be correctly classified by the edited case-base built so far. CNN makes multiple passes through the original case-base until no more additions can be made.

CNN tends to select cases that lie on, or close to, class boundaries and has been shown to make significant case-base reductions in many domains. However, while CNN performs remarkably well (considering its simplicity) it does suffer from serious problems. It is sensitive to noise, and tends to preserve noisy cases, incorrectly viewing them as important exceptions. Therefore, when applied to

noisy data-sets it can produce an edited case-base with poor classification accuracy (compared to the original case-base).

CNN is also sensitive to the order in which cases are examined and tends to preserve redundant cases during the early editing stages. For example, a case that cannot be classified by a small early-stage edited case-base will be preserved, but may turn out to be unnecessary in the light of the final edited case-base. Gates [8] described the Reduced Nearest Neighbour (RNN) method as an adaptation of CNN to cope with this problem. RNN adds a post-processing step to CNN, which attempts to contract the final edited case-base by identifying and deleting redundant cases that were added too early on - cases that can be removed from the edited case-base without reducing its classification accuracy over the original case-base.

Tomek [17] describes another improvement to CNN by ordering cases prior to editing so that redundant cases tend to be examined late in the editing process when they are unlikely to be mistaken as useful. Tomek suggests the use of the distance to a case's nearest neighbour in an opposing class (nearest unlike neighbour, or NUN, distance) as an ordering function. Cases with small NUN distances are examined first as they are likely to lie on class boundaries and therefore have a significant classification competence (see also [7]). The CNN-NUN variation has been shown to work well in that it significantly reduces edited case-base size, but still suffers from noise problems.

## 2.2 The ENN Family

The Edited Nearest Neighbour (ENN) technique is a perfect counterpoint to the CNN family [18]. ENN is a noise removal technique. Unlike CNN, which builds an edited case-base from scratch by adding cases from the original case-base, ENN contracts the original case-base by deleting cases, to produce the final edited case-base. A case is deleted if it is incorrectly classified by its  $k$  nearest neighbours.

Thus, ENN deletes cases that appear as exceptions within regions of the classification space containing cases of the same class (essentially the opposite of CNN). ENN works well to remove noise from case-base, and can produce edited case-bases with higher classification accuracy than the original case-base. However, in most domains it results in only minor size reductions, since noise usually accounts for only a small fraction of the training cases.

Two variations of ENN are described by Tomek [17]. Repeated Wilson Editing (RENN) repeatedly applies ENN to the case-base until no more cases can be deleted. This makes it possible to delete small clusters of noisy cases, but of course there is the risk that these clusters represent genuine exceptions. The All-kNN approach adds one further twist, by increasing the value of  $k$  for each iteration of RENN so that more subtle examples of noise can be detected and removed. Again these techniques can improve classification accuracy but do not result in major reductions in the original case-base size.

### 2.3 The DROP Family

Wilson and Martinez [19] present three editing algorithms that are guided by two sets for each case: the  $k$  *nearest neighbours* and the *associates* of the case. The associates of a case  $c$  are those cases which have  $c$  as one of their nearest neighbours. The algorithms reduce a case-base by deleting cases to produce an edited set. A case  $c$  is removed if, after removal, at least as many of its associates can be correctly classified.

DROP1 is the simplest member of the family and applies the above deletion criterion to each case in turn, updating the nearest neighbour and associate sets for the remaining cases after each deletion. DROP1 tends to remove noise from the original case-base, as deleting a noisy case will usually result in an increase in the classification of its neighbours.

DROP2 is identical to the DROP1 algorithm except that cases are sorted in descending order of NUN distance (cf. CNN-NUN previously) prior to deletion. Therefore, cases in the interior of class regions are deleted before cases at the boundaries. DROP2 also differs from DROP1 in that there is no updating of the associates sets after deletion. Thus, a case can have associates that have already been deleted, but which can help to guide further deletions - the intuition is that more information is available to make more informed judgements during the later stages of deletion.

Finally, the DROP3 procedure is a hybrid editing technique in that it combines an ENN pre-processing stage with the DROP2 procedure so that noisy cases are removed prior to DROP2 editing. DROP3 performs extremely well in terms of final edited case-base size and classification accuracy and Wilson and Martinez [19] report that DROP3 is one of the best available instance-based classifiers over wide variety of data-sets and classification tasks.

### 2.4 Footprint Deletion and the Iterative Case Filtering Algorithm

Smyth and Keane [12] describe a competence-guided case deletion method called Footprint Deletion for removing redundant cases from case-based reasoning systems. Footprint Deletion is guided by estimates of the competence of cases (with respect to problem solving). Two important sets are defined: the *reachability set* of a case  $c$  is the set of cases that can successfully solve  $c$ ; the *coverage set* of a case  $c$  is the set of cases that  $c$  can solve (see Defs. 1 and 2). The size and contents of these sets represent the local competences of a case.

$$ReachabilitySet(c) = \{c' \in C : Solves(c', c)\} \quad (1)$$

$$CoverageSet(c) = \{c' \in C : Solves(c, c')\} \quad (2)$$

Smyth and Keane explain how these local competence estimates can be used directly to categorise the global competence contributions of individual cases, from critical to non-critical. Their Footprint Deletion policy uses these categories to order cases for deletion, ensuring that critical cases are never removed where

possible. They demonstrate that the technique is particularly effective in many case-based reasoning systems, guaranteeing competence preservation where more traditional case deletion techniques result in competence degradation (see [12] for more information).

Brighton [3] describes how the reachability and coverage sets of cases can be adapted to estimate the local competences of cases in case-based classifiers. The reachability set of a case  $c$  is the set of cases that can be used to correctly classify  $c$ , that is, the set of cases that are closer to  $c$  than  $c$ 's nearest unlike neighbour. Similarly, the coverage set of a case  $c$  is the set of cases that  $c$  can correctly classify, that is, the set of cases for which  $c$  is a nearest neighbour of the same class.

The analogy between these sets and the Wilson and Martinez nearest neighbour and associate sets should be clear. One important difference, however, is that while the nearest neighbour set is always a fixed size,  $k$ , reachability sets are bounded by the nearest unlike neighbour of a case. This property is used to good effect by Brighton and Mellish [4] in their Iterative Case Filtering (ICF) technique. ICF is a case selection algorithm that contracts the original case-base to a reduced edited case-base by deleting cases whose reachability set size is larger than their coverage set size. In other words, delete a case  $c$  if  $c$  is solved by more cases that it can itself solve. ICF makes multiple passes over the original case-base, updating the reachability and coverage sets of the remaining cases after each pass, and terminating when no more cases can be deleted. Like DROP3, ICF employs an ENN pre-processing stage to remove noisy cases from the original case-base prior to editing.

The ICF method is an excellent performer on a wide range of standard datasets, and, in terms of edited case-base size and classification accuracy, Brighton and Mellish show that it has comparable performance to DROP3, the best of Wilson and Martinez's algorithms.

### 3 Competence-Guided Case-Base Editing

Recently editing solutions have taken a new direction in the sense that they are guided by explicit models of competence, originating from work by the CBR community [12,13] (see Section 2.4). The innovation here is two-fold: first, that it is possible to usefully model the competence characteristics of individual cases or cases, and second that an effective competence model can act as a useful case selection guide.

In previous work [14] we have reported early-stage results for a technique related to the one described below in Section 3.2. In this paper we extend this earlier work by describing three new competence-guided editing techniques along with a comprehensive evaluation on a range of classification problems, and compared to a variety of existing editing approaches. However, before presenting this evaluation (Section 4), in this section we outline our three competence-guided editing methods and the competence metrics that they use.

### 3.1 A Framework for Competence-Guided Editing

Conventional editing algorithms operate in one of two ways: either an edited case-base is built from scratch by adding cases to it from the original set; or an edited case-base is produced by deleting cases from the original set. In this section we describe an algorithm that combines these two strategies and that leads to three different competence-guided editing methods, each using a different type of competence metric.

```

T:      Original training cases
CM:     Competence Model
Eval:   Ordering function (RFC/MCOV/RC)

1  Edit(T, CM, Eval)
2    R-Set  $\leftarrow$  RENN(T) {that is, repeated ENN}
3    E-Set  $\leftarrow$  {}
4    While R-Set  $\neq$  {}
5      c  $\leftarrow$  Next case in R-Set according to Eval
6      E-Set  $\leftarrow$  E-Set  $\cup$  {c}
7      R-Set  $\leftarrow$  R-Set - CoverageSet(c)
8      Update(CM)
9    EndWhile
10   Return (E-Set)

```

**Fig. 1.** The basic algorithm template of our competence-guided editing techniques. Each algorithm will differ according to the evaluation policy

The basic algorithm is shown in Figure 1. Cases are selected for addition to the edited case-base using one of three evaluation policies, each of which is described in the following section. Once a case  $c$  is selected for the edited case-base (E-Set), all of the cases that it can solve (that is, its coverage set) are removed from the remaining cases (the R-Set), and the reachability and coverage sets of these remaining cases are updated. This procedure is repeated until there are no more remaining cases.

Unlike other editing methods, this algorithm explicitly deletes groups of cases (the coverage set of a selected case) from further consideration (line 7) as a side effect of each addition operation (line 6). Because of this, the algorithm tends to converge rapidly on a final edited case-base.

### 3.2 Evaluation Policies

The order in which cases are considered during editing can have an important bearing on the quality of the final edited case-base. For example, we saw earlier that CNN suffers because its selection rule cannot distinguish between useful and redundant cases during the early stages of editing. One solution is to order cases

prior to editing, using an evaluation function to measure the expected usefulness of a case - the next case to be considered during editing can then be the most useful case left in the original case-base. We propose three competence-guided evaluation policies that measure different competence properties of cases.

**Reach For Cover (RFC)** The size of the reachability set of a case can be used as an estimate of how difficult this case is to solve, and thus how important this case is likely to be with respect to classification accuracy. For example, a case with a very small reachability set cannot be solved by many other cases (perhaps even no other cases) and so this case may be a crucial element of any edited set if it represents a set of future target cases. The RFC evaluation function implements this idea: the usefulness of a case is an inverse function of its reachability set size. Cases with small reachability sets are considered for membership in the edited case-base before cases with large reachability sets. Cases with small reachability sets tend to occur at the boundaries of classes and so using RFC will lead to the preservation of boundary cases and the deletion of interior cases. Ordinarily this would make the RFC evaluation policy sensitive to noise, but the RENN noise-filtering stage at the start of the editing template will compensate for this.

**Maximal Cover (MCOV)** The size of the coverage set of a case can also be used as an evaluation criterion. Cases with large coverage sets can classify many target cases and as such must make a significant contribution to classification competence. These cases should be preserved in any edited case-base. The MCOV evaluation function implements this idea by preferring cases with large coverage sets. This evaluation policy will tend to preserve cases from the interiors of class clusters, since these cases will have the largest coverage sets - cases near the boundary of a class will only correctly classify cases on one side of the boundary.

**Relative Cover (RC)** The RFC and MCOV functions, and the ICF method discussed in the previous section, use local competence measures only to guide the editing process. The coverage and reachability sets of a case encode local competence characteristics, and tell us very little about a case's global competence characteristics. For example, a case  $c$  may have a large coverage set, and according to MCOV it should be a prime candidate for case selection because it must make a significant contribution to classification competence. However, if the cases that  $c$  covers are themselves covered by other cases then the unique competence contribution of  $c$  is reduced. What is needed is a metric for computing the competence of a case relative to other nearby (interacting) cases. Relative coverage is just such a metric (see Def 3). It is based on the idea that if a case  $c$  is covered by  $n$  other cases then each of the  $n$  cases will receive a contribution of  $1/n$  from  $c$  to their relative coverage measures (this competence metric was first introduced in [14] and the RC editing method proposed here is a variation of the one described in that paper).

$$RelativeCoverage(c) = \sum_{c' \in CoverageSet(c)} \frac{1}{|ReachabilitySet(c')|} \quad (3)$$

Cases with high relative coverage tend to cover hard to solve (classify) cases, and thus RC will tend to bias cases that lie near to class boundaries - it will actually lead to the preservation of cases that can be used to solve the boundary cases of a class (for more information see [14]).

## 4 Experimental Evaluation

In this section we evaluate the performance of these new competence-guided techniques on a range of data-sets, under various noise conditions, and by comparison to a number of traditional and state-of-the-art editing techniques.

### 4.1 Experimental Set-Up

We use 9 data-sets from the UCI ML Repository [2]. A random 20% of the cases in each data-set are reserved as unseen targets for testing classification accuracy and the remaining 80% are used as training data. This is repeated 30 times to produce 30 training and test sets for each data-set. We further choose four representative editing methods to compare against our three new techniques: (1) CNN - to facilitate a comparison with a pure redundancy reduction technique; (2) RENN - to facilitate a comparison with a pure noise removal technique. (3) ICF - to facilitate a comparison with a state-of-the-art hybrid technique combining noise removal and redundancy reduction; and (4) DROP3 - to facilitate a comparison with one of the *best-of-the-best* editing algorithms currently available.

### 4.2 Size vs. Competence

**Method:** For each case editing algorithm and data-set we compute the mean edited case-base size (as a percentage of the original case-base size) and the mean classification accuracy (over the appropriate test sets). The mean benchmark (BM) accuracy (that is, the average accuracy of the unedited original case-bases) is also noted. Obviously size of the benchmark case-base is 100% as it includes all of the original cases.

**Results:** The results are tabulated and averaged in Table 1 alongside the mean benchmark accuracy and size values - within each table cell there are two values, the top one represents the mean edited case-base size as a percentage of the original case-base, and the bottom one represents the mean percentage accuracy of the edited case-bases over the unseen target problems.

**Discussion:** On average the new methods are seen to out-perform CNN, RENN and ICF in both edited case-base size and accuracy; all except MCOV beat RENN terms of size and accuracy. It is no surprise that significant size



improvements are seen for the new methods over CNN and RENN (given that CNN and RENN engage in redundancy reduction or noise removal, but not both). Each of our techniques also out-performs DROP3 with respect to size. However, there are minor accuracy reductions.

Overall, the RC method is the outright winner in terms of edited case-base size. It produces edited case-bases that are nearly half the size of the ICF and DROP3 case-bases. RC also manages to outperform ICF in terms of average accuracy with only a minor accuracy reduction in comparison to DROP3 (0.05%). In fact, it produces significantly smaller edited case-bases than ICF and DROP3 in every one of the tested data-sets, and at least as high accuracy values as ICF in 7 out of the 9 data-sets.

Earlier we predicted that the ICF, RFC, MCOV techniques would suffer because of their reliance on local competence estimates as a guide for case selection. These local estimates cannot account for the interactions between the competence contributions of related cases and therefore should limit the effectiveness of these methods. RC was forwarded as a potential solution and this is confirmed by our results - the more sophisticated relative coverage metric yields smaller edited case-bases without compromising accuracy.

**Table 1.** Size and accuracy results for all data-sets and algorithms. The number under each data-set name is the size of the original training set (this represents 80% of the data-set, which is used as training data in these experiments)

| Algorithm/<br>Data set | -BM-         | -CNN-          | -RENN-         | DROP3-         | -ICF-          | -RFC-          | -MCOV-         | -RC-           |
|------------------------|--------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Adult<br>(329)         | 100<br>75.69 | 42.85<br>71.91 | 73.71<br>77.28 | 10.17<br>77.36 | 13.16<br>76.75 | 12.02<br>77.97 | 6.49<br>75.45  | 6.353<br>75.28 |
| BC<br>(560)            | 100<br>95.01 | 11.67<br>92.88 | 96.36<br>96.75 | 3.24<br>96.15  | 3.33<br>95.73  | 2.63<br>96.01  | 2.34<br>96.13  | 2.12<br>95.91  |
| Bupa<br>(276)          | 100<br>62.9  | 57.03<br>61.88 | 58.37<br>63.57 | 22.91<br>64.11 | 15.98<br>63.96 | 12.38<br>63.48 | 10.46<br>64.25 | 7.97<br>62.94  |
| Credit<br>(552)        | 100<br>82.39 | 38.64<br>76.69 | 83.74<br>84.88 | 15.28<br>83.04 | 13.85<br>83.41 | 13.83<br>83.96 | 11.33<br>83.94 | 10.68<br>84.28 |
| Cylinder<br>(432)      | 100<br>79.51 | 44.07<br>75.56 | 69.51<br>71.7  | 22.2<br>71.11  | 17.79<br>70.9  | 13.67<br>71.85 | 11.68<br>72.04 | 11.2<br>71.17  |
| Diabetes<br>(244)      | 100<br>60.78 | 59.03<br>60.06 | 56.53<br>59.83 | 18.02<br>62.89 | 14.29<br>60.56 | 10.1<br>63.17  | 8.66<br>62.61  | 8.28<br>64.22  |
| Hepatitis<br>(124)     | 100<br>77.85 | 39.81<br>73.98 | 74.41<br>81.08 | 7.82<br>82.14  | 13.06<br>80.54 | 8.28<br>79.68  | 4.73<br>81.29  | 4.41<br>81.72  |
| Hungarian<br>(236)     | 100<br>78.16 | 37.84<br>75    | 78.36<br>81.72 | 9.92<br>80.92  | 8.50<br>79.71  | 9.10<br>81.32  | 7.59<br>79.71  | 7.40<br>80.92  |
| Voting<br>(348)        | 100<br>91.61 | 16.88<br>90.65 | 91.05<br>92.57 | 7.65<br>93.56  | 9.73<br>91.69  | 5.56<br>92.64  | 4.25<br>92.07  | 4.14<br>92.61  |
| Average<br>344.55      | 100<br>78.21 | 38.65<br>75.40 | 75.78<br>78.82 | 13.02<br>79.02 | 12.19<br>78.14 | 9.73<br>78.90  | 7.51<br>78.61  | 7.16<br>78.97  |

### 4.3 Editing Noisy Data

**Method:** As with the previous experiment, for each algorithm and data set, an edited case-base is produced, and the average size and accuracy is noted. However, this time we introduce class-noise into the case data (by flipping the class values of randomly selected cases to incorrect class values) to test how each algorithm performs under noisy conditions. For example, 10% noise means that 10% of the cases have their classes flipped to incorrect values. Of course there will inevitably be some noise present in the case-bases to begin with, so the actual percentage of real noise in our training cases will be greater than the stated amount.

**Results:** Table 2 shows the average size and accuracy results over all data sets, for each algorithm, and for the various levels of noise ranging from 0% to 30%. Once again, the top value in each cell is the percentage edited case-base size, and the bottom value is the percentage classification accuracy.

**Table 2.** Overall size and accuracy results for each algorithm under different noise conditions

| Algorithm/<br>Noise | -BM-  | -CNN- | -RENN- | -DROP3- | -ICF- | -RFC- | -MCOV- | -RC-  |
|---------------------|-------|-------|--------|---------|-------|-------|--------|-------|
| 0%                  | 100   | 38.65 | 75.78  | 13.02   | 12.19 | 9.73  | 7.51   | 7.16  |
|                     | 78.21 | 75.40 | 78.82  | 79.02   | 78.14 | 78.9  | 78.61  | 78.97 |
| 10%                 | 100   | 50.56 | 66.97  | 12.78   | 15.85 | 8.88  | 6.93   | 6.60  |
|                     | 76.59 | 71.83 | 78.12  | 78.16   | 77.09 | 77.91 | 77.55  | 77.58 |
| 20%                 | 100   | 58.27 | 57.36  | 13.66   | 17.57 | 8.48  | 6.58   | 6.30  |
|                     | 72.32 | 66.43 | 76.32  | 75.54   | 71.95 | 76.13 | 75.72  | 75.72 |
| 30%                 | 100   | 63.68 | 48.54  | 16.08   | 20.56 | 9.18  | 7.15   | 6.93  |
|                     | 67.22 | 60.99 | 73.37  | 71.20   | 65.18 | 73.04 | 73.31  | 73.07 |

**Discussion:** Our new techniques produce significantly smaller edited case-bases than ICF and DROP3 for all levels of noise used. They produce edited case-bases that are between 1.8 and 2.3 times smaller than the equivalent DROP3 edited case-base, and between 1.7 and 3 times smaller than the ICF edited case-bases. As expected, the results are even more dramatic when compared to the CNN and RENNN algorithms. Interestingly, the sizes of the edited sets produced by our competence-guided techniques remain relatively stable across the different noise levels, unlike the sizes of the sets produced by ICF or DROP3 (or indeed CNN and RENNN). Indeed if anything the competence-guided editing techniques are producing smaller edited case-bases as more noise is introduced, suggesting that noisy cases are being correctly identified and removed. In contrast, the ICF and DROP3 methods produce larger edited case-bases as more noise is introduced, and this suggests that they are not removing all of the noisy cases during editing; the edited sets are growing in size because noisy cases are remaining after editing. For example, the edited case-base size for ICF grows from 12.19%

at the 0% noise level to 20.56% at the 30% noise level, that is an increase of 70%, suggesting that a significant number of noisy cases are being missed.

This is borne out when we look at the accuracy of the edited case-bases. In general, as expected, there is a gradual decrease in the mean classification accuracy across all algorithms as the noise level increases. However, the rate of decrease differs greatly. For example, at noise levels greater than 10% the competence-guided editing techniques are producing edited sets that have greater classification accuracy than those produced by ICF and DROP3 (and CNN) while at the same time being significantly smaller in size. As the noise level increases, the classification accuracy drops by 13% and 8% for ICF and DROP3, respectively, but by only about 5% for our editing techniques. The reason of course is that DROP3, and ICF in particular, are retaining more noisy cases than our competence-guided techniques, and thus they are suffering from more misclassifications as a result. At the maximum 30% noise level our competence-guided editing techniques achieve greater accuracy than BM, CNN, ICF, and DROP3 but with significantly smaller case-bases (6%-9% of the original case-base). Once again, we find that the RC technique, with its more sophisticated competence model, generally outperforms the RFC and MCOV methods, as predicted.

## 5 Conclusions

In this paper we have described and evaluated three new competence-guided editing techniques under a variety of experimental conditions. The algorithms all benefit from superior performance characteristics when compared to existing editing methods. In particular, we have seen that they can produce significantly smaller edited sets than today's best editing algorithms without compromising competence. In fact, we have shown that our new techniques benefit from improved noise tolerance, so much so that they display better classification performance than all tested editing algorithms as the amount of noise in the original case-bases increases.

In this paper we have investigated how are editing techniques perform on classification data-sets. Future work will report how the techniques map on to more general CBR tasks such as synthesis tasks. Finally, we have recently shown how our competence model can also be used to develop innovative solutions to problems such as case addition, case retrieval, case-base visualization, and authoring support [13,14,15] We have also developed a new update procedure which means that these solutions are now available at a reduced cost [16].

## References

1. Aha, D. W., Kibler, D., and Albert, M. K.: *Instance-Based Learning Algorithms*, Machine Learning **6** (1991), 37–66. 186
2. Blake, C., Keogh, E., and Merz, C. J.: *UCI repository of Machine Learning Databases*, Irvine, CA: University of California. Department of Information and Computer Science, (1998). 193

3. Brighton, H.: *Information Filtering for Lazy Learning Algorithms*, Masters thesis, Centre for Cognitive Science, University of Edinburgh, Scotland, (1997). 190
4. Brighton, H and Mellish, C.: *On the Consistency of information filters for Lazy learning algorithms*, Proceedings of the 3rd European Conference on the Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science (Jan Rauch Jan M. Zytkow, ed.), Springer Verlag, (1999), pp. 283–288. 190
5. Cameron-Jones, R. M.: *Minimum Description Length Case-Based Learning.*, Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, World Scientific, (1992), pp. 368–373. 186
6. Chang, C. L.: *Finding Prototypes for Nearest Neighbour Classifiers*, IEEE Transactions on Computers **C-23** (1974), 1179–1184. 186
7. Dasarathy, D. V.: (ed.), *Nearest Neighbor Norms: NN Pattern Classification Techniques*, IEEE Press, (1991). 186, 188
8. Gates, G. W.: *The Reduced Nearest Neighbor Rule*, IEEE Transactions on Information Theory **IT-18(3)** (1972), 431–433. 186, 188
9. Hart, P. E.: *The Condensed Nearest Neighbor Rule*, IEEE Transactions on Information Theory **IT-14** (1967), 515–516. 186, 187
10. King, R. D., Feng, C. and Sutherland, A.: *Statlog: Comparison of Classification Algorithms on Large Real-Worlds Problems*, Applied Artificial Intelligence **9(3)** (1995), 289–333. 186
11. McKenna, E. and Smyth, B.: *Competence-guided Editing Methods for Lazy Learning*, Proceedings of the 14th European Conference on Artificial Intelligence, (2000). 187
12. Smyth, B. and Keane, M. T.: *Remembering to Forget: A Competence Preserving Case Deletion Policy for CBR Systems*, Proceedings of the 14th International Joint Conference on Artificial Intelligence (Chris Mellish, ed.), Morgan Kaufmann, (1995), pp. 377–382. 187, 189, 190
13. Smyth, B. and McKenna, E.: *Modelling the Competence of Case-Bases*, Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence (B. Smyth and P. Cunningham, eds.), Springer Verlag, (1998), pp. 208–220. 187, 190, 196
14. Smyth, B. and McKenna, E.: *Building Compact Competent Case-Bases*, Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence (Klaus Dieter Althoff, Ralph Bergmann, and L.Karl Branting, eds.), Springer Verlag, (1999), pp. 329–342. 187, 190, 192, 193, 196
15. Smyth, B. and McKenna, E.: *Footprint-Based Retrieval*, Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence (Klaus Dieter Althoff, Ralph Bergmann, and L.Karl Branting, eds.), Springer Verlag, (1999), pp. 343–357. 196
16. Smyth, B. and McKenna, E.: *An Efficient and Effective Procedure for Updating a Competence Model for Case-Based Reasoners*, Proceedings of the 11th European Conference on Machine Learning (Ramon Lopez de Mantaras and Enric Plaza, eds.), Springer Verlag, (2000). 196
17. Tomek, I.: *Two Modifications of CNN*, IEEE Transactions on Systems, Man, and Cybernetics **7(2)** (1976), 679–772. 186, 188
18. Wilson, D. L.: *Asymptotic Properties of Nearest Neighbor Rules Using Edited Data*, IEEE Transactions on Systems, Man, and Cybernetics **2-3** (1972), 408–421. 186, 188
19. Wilson, D. R. and Martinez, T. R.: *Instance Pruning Techniques*, Proceedings of the 14th International Conference on Machine Learning (D. Fisher, ed.), Morgan Kaufmann, (1997), pp. 403–411. 186, 189

# Intelligent Case-Authoring Support in CaseMaker-2

David McSherry

School of Information and Software Engineering, University of Ulster  
Coleraine BT52 1SA, Northern Ireland  
dmg.mcsherry@ulst.ac.uk

**Abstract.** CaseMaker is an interactive environment for intelligent case-authoring support in CREST, a case-based reasoner for estimation tasks, in which the selection of cases for addition to a case library is guided by empirical evaluation of the coverage contributions of candidate cases. We present a new version of the environment called CaseMaker-2 which is designed to support case authoring more effectively by eliminating a previous requirement for the evaluation of candidate cases to be repeated following the addition of a new case to the library. A key role in the approach is played by *eValueate*, an algorithm for dynamic partitioning of the space of uncovered cases in such a way that the partition containing a given case represents the minimum additional coverage provided by its addition to the case library.

## 1 Introduction

The development of tools and techniques to support case-base maintenance has recently attracted significant research interest in case-based reasoning (CBR). Strategies for case addition, deletion, and refinement have been proposed as solutions to the problems caused by decreasing efficiency of retrieval as the size of a case library increases [1,2], inadequate coverage [3], inconsistent or redundant cases [4], and non-compliance with design guidelines [5]. Other promising developments in case-base maintenance include updating of case indexes through introspective learning [6] and rule-based updating of cases in response to the maintenance challenges presented by rapidly-changing domains [7].

While the emphasis in maintenance tends to be on improving performance through remedial measures, our focus in this paper is on the *prevention* of problems like redundancy and inadequate coverage through intelligent support for case authoring. A basic requirement in case-authoring support is evaluation of the coverage provided by a given case library. Smyth and McKenna [8] have developed a competence model that enables coverage to be accurately estimated provided the case library is representative. A case-authoring tool based on the model enables coverage to be visualised, so that authors can concentrate on areas of low coverage. Adopting instead an empirical approach to the evaluation of coverage, we have presented an algorithm called disCover [9] for the discovery of all cases that can be solved from a given case library in CREST, a CBR tool for estimation tasks [10,11]. The

algorithm has been adapted as a tool for identifying the most useful case to be added to a case library; that is, the case that will provide maximum additional coverage. The result is CaseMaker, an environment for intelligent case-authoring support in which the selection of cases for addition to a case library is guided by the coverage contributions of candidate cases [12].

A limitation of our previous approach arises from the fact that the coverage contributions of still-uncovered cases may change following the addition of a new case to the case library. Their evaluation must therefore be repeated by CaseMaker in order to identify the next most useful case to be added to the case library. In this paper, we present a new approach to the evaluation of uncovered cases that enables their minimum coverage contributions to be dynamically updated as the case library is extended. A key role in the approach is played by an algorithm for dynamic partitioning of the space of uncovered cases in such a way that the partition containing a given case represents the minimum additional coverage provided by its addition to the case library.

Section 2 describes the adaptation and retrieval strategies used in CREST and its demand-driven approach to the discovery of adaptation knowledge. In Section 3, we present a new version of CaseMaker called CaseMaker-2 and the techniques it uses in the discovery and updating of knowledge to support case authoring. In Section 4, an example case library in the domain of property valuation is used to illustrate the support for case authoring provided by the new version of CaseMaker.

## 2 Case-Based Reasoning in CREST

A case is represented in CREST as a tuple  $C = (a_1, a_2, \dots, a_n)$ , where  $a_1, a_2, \dots, a_n$  are the values of the case attributes  $x_1, x_2, \dots, x_n$  for  $C$ . The target problem space is the Cartesian product of the (finite) domains of the case attributes. A library case is one that is stored, with its known solution, in a case library. A case that is not currently in the case library is called a non-library case. Given a target case  $C$ , the task of the case-based reasoner is to estimate its value,  $val(C)$ , from the values of existing cases in the case library. An innovative feature of CREST is its demand-driven approach to the discovery of adaptation knowledge [11]. In the absence of an exact match for a target case, its solution is based on three library cases, one retrieved for its similarity to the target case and the others to provide the knowledge required to adapt the similar case. The triple of retrieved cases is called an *adaptation triple* for the target case.

**Definition 1** A triple  $C_1, C_2, C_3$  of cases is called an *adaptation triple* for a target case  $C_t$  if there exist  $C_s, C_a, C_b \in \{C_1, C_2, C_3\}$  such that  $C_t$  and  $C_s$  differ only in the value of a single attribute,  $C_a$  and  $C_b$  differ only in the value of the same attribute, and the differing attribute has the same value for  $C_t$  and  $C_a$  and the same value for  $C_s$  and  $C_b$ . If  $C_s = C_1$ ,  $C_a = C_2$ , and  $C_b = C_3$  then  $C_1, C_2, C_3$  is said to be an *ordered adaptation triple* for  $C_t$ .

The heuristic most often used to adapt the similar case, according to which  $val(C_t)$

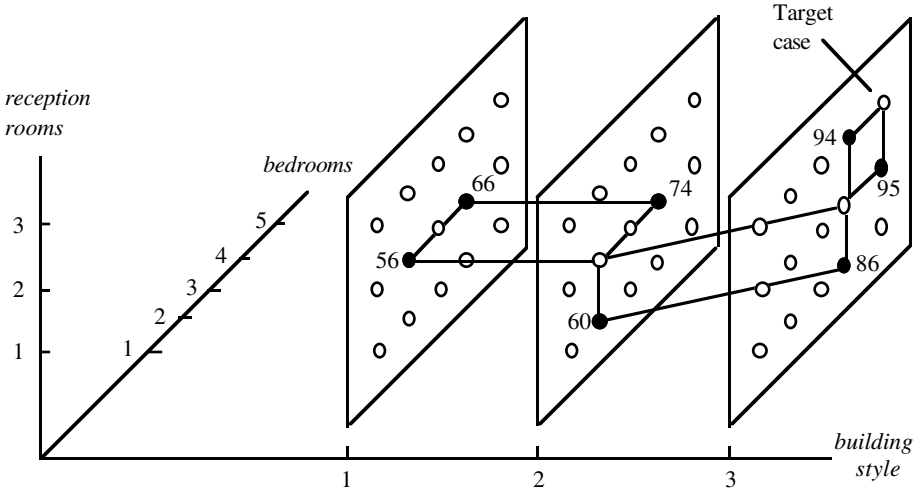


**Fig. 1.** Recursive adaptation in CREST

$= val(C_1) + val(C_2) - val(C_3)$  for any case  $C_t$  and ordered adaptation triple  $C_1, C_2, C_3$  for  $C_t$  is called the *difference heuristic* [10]. It always gives the correct value for a target case provided the value of a case is an additive function of its attributes and the values of library cases are known without error. Recursive adaptation has been shown to increase coverage significantly [11] even when restricted, in the interest of retrieval efficiency, to adaptation triples in which at most one case is a non-library case.

**Definition 2** An adaptation triple  $C_1, C_2, C_3$  for a target case  $C_t$  is admissible if it includes at most one non-library case.

The filled circles in Fig. 1 represent 7 cases in an example library in the domain of property valuation. The cases are residential properties and their attributes are building style (1, 2 or 3), no. of bedrooms (1 to 5), and no. of reception rooms (1, 2 or 3). The values of the library cases according to a hypothetical function of their attributes are shown in units of £1,000. The solution of a target case by recursive adaptation is also illustrated. An ordered adaptation triple for the target case  $C_t = (3,5,3)$  is provided by the library cases  $C_1 = (3,4,3)$ ,  $C_2 = (3,5,2)$  and the non-library case  $C_3 = (3,4,2)$ . An ordered adaptation triple for  $(3,4,2)$  is in turn provided by  $(3,4,1)$ ,  $(2,2,2)$  and  $(2,2,1)$ . Finally, an ordered adaptation triple for  $(2,2,2)$  is provided by  $(1,2,2)$ ,  $(2,4,2)$  and  $(1,4,2)$ . According to the difference heuristic, an estimate for the value of the target case is therefore:

$$\begin{aligned}
 val(3,5,3) &= val(3,4,3) + val(3,5,2) - val(3,4,2) \\
 &= 94 + 95 - (val(3,4,1) + val(2,2,2) - val(2,2,1)) \\
 &= 94 + 95 - 86 - (val(1,2,2) + val(2,4,2) - val(1,4,2)) + 60 \\
 &= 94 + 95 - 86 - 56 - 74 + 66 + 60 = 99
 \end{aligned}$$

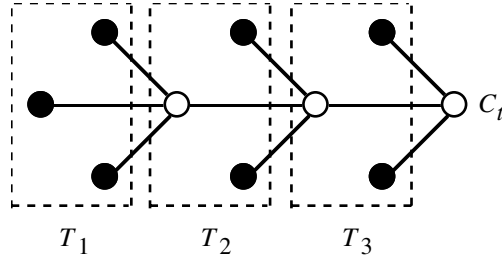


**Fig. 2.** Solution of a target case  $C_t$  based on a sequence of three adaptation triples

**Definition 3** A case  $C$  is covered by a given case library  $L$  if  $C \in L$  or  $C$  can be solved by adaptation based on admissible adaptation triples.

Given a case library  $L$ , we denote by  $\bar{L}$  the set of all cases that are covered by  $L$ . Defined as the percentage of cases in the problem space that are covered by a given case library, coverage of the  $3 \times 5 \times 3$  problem space by the example case library is 60% (or 27 cases). As we show in Section 4, full coverage of the problem space can be achieved by adding just two more cases to the case library. Of course, not any pair of cases will do; indeed, it is possible to add 10 new cases before full coverage is achieved. The fact that the sample size required for full coverage ranges from 20% (9 cases) to 38% (17 cases) illustrates the need for intelligent support for case authoring.

**Proposition 1** A non-library case  $C_t$  is covered if and only if there exists a sequence  $T_1, T_2, \dots, T_n$  of admissible adaptation triples such that  $T_1$  consists only of library cases,  $T_n$  is an adaptation triple for  $C_t$  and if  $n > 1$ , then for  $1 \leq i \leq n-1$ ,  $T_i$  is an adaptation triple for the non-library case in  $T_{i+1}$ . Such a sequence of adaptation triples (which may not be unique) is said to cover  $C_t$ .

The solution of a target case  $C_t$  by recursive adaptation involving a sequence of adaptation triples  $T_1, T_2, T_3$  is illustrated for a problem space of arbitrary dimensions in Fig. 2. The filled circles represent library cases. The central placing of the non-library case in each adaptation triple is merely a notational convenience and has no bearing on its role in the adaptation triple. In disCover, the *reversibility* of adaptation triples [9] plays an important role in the discovery of all cases that can be solved from a given case library  $L$ .

**Proposition 2** If  $C_1, C_2, C_3$  is an adaptation triple for  $C_t$ , then  $C_t, C_2, C_3$  is an adaptation triple for  $C_1$ . Similarly,  $C_t, C_1, C_3$  is an adaptation triple for  $C_2$ , and  $C_t, C_1, C_2$  is an adaptation triple for  $C_3$ .

The reversibility of adaptation triples (which is independent of the heuristic used for adaptation) is illustrated in Fig. 3. The left-hand side shows an adaptation triple  $C_1, C_2, C_3$  for a target case  $C_t$ . One of the cases in the adaptation triple,  $C_2$ , is a non-library case. In the other half of the diagram, the roles of  $C_t$  and  $C_2$  are reversed, with  $C_t$  now acting as one of the cases in an adaptation triple for  $C_2$ .



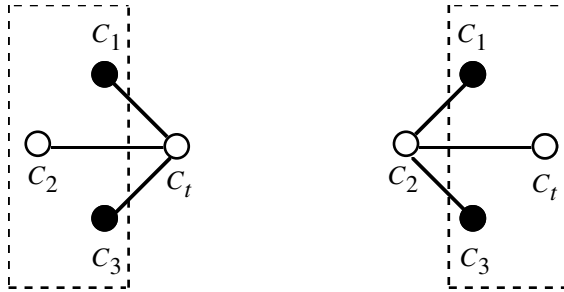


Fig. 3. Reversibility of adaptation triples

### 3 The CaseMaker-2 Environment

CaseMaker-2 is a collection of tools for the discovery, application, and updating of knowledge to support case authoring, such as the minimum coverage contribution of each case that is not covered by a given case library. Two of its main tasks are:

- disCover*: discover all cases that can be solved from a given case library
- eEvaluate*: partition the space of uncovered cases so that the partition containing a given case represents the minimum additional coverage provided by its addition to the case library

The *disCover* and *eEvaluate* tasks need only be performed once, either before any cases are added to a case library (when they are trivial) or when CaseMaker-2 is first applied to an existing case library. Thereafter, CaseMaker-2 simply repeats a *reRecommend* - *reClaim* - *rePartition* cycle as new cases are added to the case library. The tasks in the cycle are:

- reRecommend*: identify and recommend the most useful case(s) to be added to the case library
- reClaim*: identify all previously uncovered cases that are now covered following the addition of a new case to the case library
- rePartition*: update the partitions of still-uncovered cases following the addition of a new case to the case library

#### 3.1 Reversing the Direction of CBR

As the following proposition shows, an admissible adaptation triple for a case that is covered by a given case library  $L$  can contain only covered cases. The significance of this result is that it enables the direction of CBR to be reversed in *disCover*'s search for cases that can be solved from a given case library. Given a covered case  $C_d$ , *disCover* searches for admissible adaptation triples for  $C_d$  that include non-library cases not already known to be covered. By Proposition 3, the non-library case in any

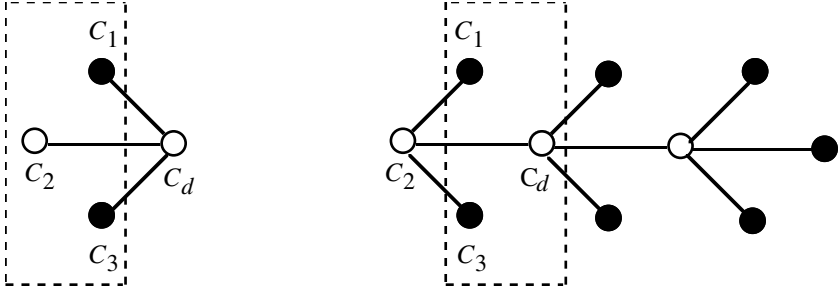


Fig. 4. A non-library case in an adaptation triple for a covered case must itself be covered

such adaptation triple can be added to the list of covered cases. The algorithm repeatedly processes a set of *active* cases, initially the set of all library cases. In each new cycle, the active cases are those that were discovered in the previous cycle. The discovery process continues until the list of active cases is empty.

**Proposition 3** *If  $C_d \in \bar{L}$  and  $C_1, C_2, C_3$  is an admissible adaptation triple for  $C_d$ , then  $C_1, C_2, C_3 \in \bar{L}$ .*

**Proof** Since an admissible adaptation triple includes at most one non-library case, it suffices to show that the non-library case, if any, is covered by  $L$ . For example, if  $C_2$  is the non-library case, then by the reversibility of adaptation triples,  $C_d, C_1, C_3$  is an adaptation triple for  $C_2$ . As  $C_d \in \bar{L}$ , either  $C_d \in L$ , in which case  $C_2$  is covered by  $C_d, C_1, C_3$ , or there exists a sequence of admissible adaptation triples that covers  $C_d$ . As Fig. 4 illustrates, any such sequence of adaptation triples can be extended to provide a sequence of adaptation triples that covers  $C_2$ . So  $C_2 \in \bar{L}$  as required.

### 3.2 Partitioning the Space of Uncovered Cases

Given a problem space  $P$  and case library  $L$ , the required partitions of the space of uncovered cases are provided by the *equivalence classes* corresponding to an equivalence relation that we now define among cases that are not covered by  $L$ .

**Definition 4** *For any  $C_x, C_y \in P - \bar{L}$ , we define  $C_x \mathbf{R}_L C_y$  if and only if  $C_x = C_y$  or there exists a sequence  $T_1, T_2, \dots, T_n$  of admissible adaptation triples such that:*

- a. *for  $1 \leq i \leq n$ ,  $T_i$  includes an uncovered case*
- b.  *$C_x$  is the uncovered case in  $T_1$*
- c.  *$T_n$  is an adaptation triple for  $C_y$*
- d. *if  $n > 1$ , then for  $1 \leq i \leq n-1$ ,  $T_i$  is an adaptation triple for the uncovered case in  $T_{i+1}$*

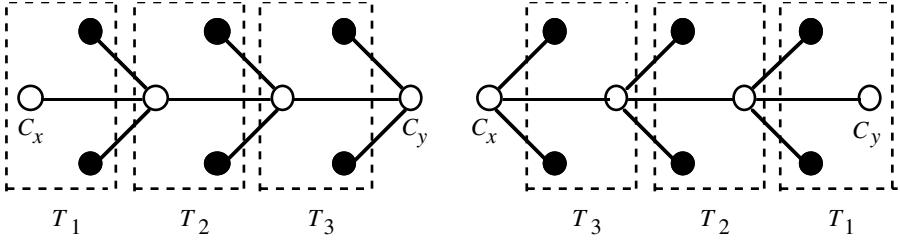


**Fig. 5.** A sequence of adaptation triples from  $C_x$  to  $C_y$  can be reversed to provide a sequence of adaptation triples from  $C_y$  to  $C_x$

**Proposition 4** For any problem space  $P$  and case library  $L$ ,  $R_L$  is an equivalence relation.

**Proof**  $R_L$  is reflexive by definition. As Fig. 5 illustrates, its symmetry follows from the reversibility of adaptation triples. It can be seen to be transitive from the fact that a sequence of adaptation triples from  $C_x$  to  $C_y$  and another from  $C_y$  to  $C_z$  can be joined to provide a sequence of adaptation triples from  $C_x$  to  $C_z$ .

The relevance of  $R_L$  in the evaluation of candidate cases for addition to a case library can be seen from the following proposition.

**Proposition 5** For any problem space  $P$ , case library  $L$ , and  $C_x, C_y \in P - \bar{L}$  such that  $C_x R_L C_y$ ,  $C_y \in \overline{LU\{C_x\}}$  and  $C_x \in \overline{LU\{C_y\}}$ .

**Proof** If  $C_x R_L C_y$ , then the addition of  $C_x$  to the case library transforms the sequence of adaptation triples that must exist between  $C_x$  and  $C_y$  to a sequence of adaptation triples that covers  $C_y$ . By the symmetry of  $R_L$ , the addition of  $C_y$  to the case library similarly ensures that  $C_x$  is covered.

It follows from Proposition 5 that for any  $C_x \in P - \bar{L}$ ,  $\overline{LU\{C_x\}} \supseteq \bar{L} \cup E_x$ , where  $E_x = \{C_y \in P - \bar{L} : C_x R_L C_y\}$  is the  $R_L$  equivalence class containing  $C_x$ .

However, it is not necessarily the case that  $\overline{LU\{C_x\}} = \bar{L} \cup E_x$ . Fig. 6 illustrates how further coverage may be gained when  $C_x$  is added to the case library. In the example shown,  $C_x$  and  $C_y$  are unrelated by  $R_L$ , and therefore have disjoint equivalence classes. When  $C_x$  is added to the case library, all cases in  $E_x$  are now covered. However, there is now an adaptation triple for  $C_y$  in which only one case is a non-library case, namely the previously covered case  $C_d$ . By Proposition 1, there must be a sequence of admissible adaptation triples that covers  $C_d$ . The admissible adaptation triple that now exists for  $C_y$  can be used to extend this sequence of adaptation triples to provide a sequence that covers  $C_y$ . Moreover, for any  $C_z \in E_y$ , the sequence of adaptation triples that now covers  $C_y$  and the sequence of adaptation

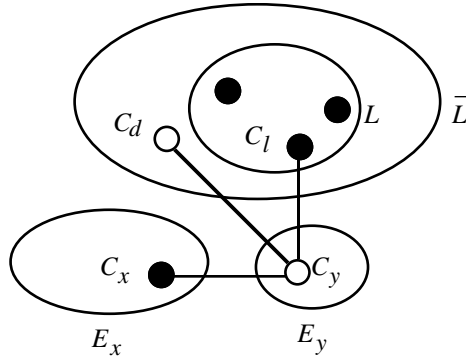


**Fig. 6.** The additional coverage provided by a new library case  $C_x$  includes but is not necessarily limited to  $E_x$

triples that must exist between  $C_y$  and  $C_z$  can be joined to provide a sequence that covers  $C_z$ . So not only  $C_y$ , but all cases in  $E_y$ , are now covered.

Thus the  $R_L$  equivalence class containing an uncovered case  $C_x$  represents the *minimum* additional coverage provided by its addition to the case library. The algorithm used by CaseMaker-2 to partition the space of uncovered cases is outlined in Fig. 7.

### 3.3 The ReCommend-ReClaim-RePartition Cycle

ReCommend's task is simply a matter of selecting an uncovered case  $C_x$  for which  $card(E_x)$  is maximum and recommending any of the cases in  $E_x$ , whose minimum coverage contributions are the same, for addition to the case library. Following the addition of a recommended case to the case library, the first of two maintenance tasks to be performed by CaseMaker-2 is the reclamation of previously uncovered cases that can now be solved.

ReClaim first adds all cases in the  $R_L$  equivalence class that contains the new case, say  $C_x$ , to the list of covered cases. To discover any additional cases that are covered by  $L \cup \{C_x\}$ , it need only consider adaptation triples for  $C_x$ , if any, that include a library case  $C_l$ , an already covered case  $C_d$ , and a previously uncovered case  $C_y$  that is unrelated to  $C_x$ . If such an adaptation triple exists, then by the reversibility of adaptation triples,  $C_x, C_l, C_d$  is an adaptation triple for  $C_y$  in which  $C_d$  is the only non-library case. As shown in Section 3.2, it follows that  $C_y$ , and all cases in its  $R_L$  equivalence class, are now covered.

The second maintenance task that follows the addition of a new case to the case library is updating the partitions of still-uncovered cases to reflect changes in their coverage contributions. When a new case  $C$  is added to the case library, the equivalence classes for  $R_{L \cup \{C\}}$  can be identified from the equivalence classes for  $R_L$  much more efficiently than building them from scratch. Before discussing the

**Algorithm** eValue( $L$ )

create a list of open cases initially consisting of all cases that are not covered by  $L$

**repeat**

select any case  $C_x$  from the list of open cases

remove  $C_x$  from the list of open cases

create a list of active cases initially containing only  $C_x$

create a list of cases related to  $C_x$  initially containing only  $C_x$

create an empty list of discovered cases

**repeat**

**for** each active case  $C_a$  and admissible adaptation triple for  $C_a$  that includes an open case  $C_y$

**do begin**

add  $C_y$  to the list of cases related to  $C_x$

add  $C_y$  to the list of discovered cases

remove  $C_y$  from the list of open cases

**end**

copy the list of discovered cases to a new list of active cases

**until** the list of active cases is empty

**until** the list of open cases is empty

**Fig. 7.** Algorithm for partitioning the space of uncovered cases

algorithm that performs this task, we show that the new equivalence relation is weaker than its predecessor.

**Proposition 6** *If a new case  $C$  is added to a case library  $L$  in a given problem*

*space  $P$ , then for any  $C_x, C_y \in P - \overline{L \cup \{C\}}$ ,  $C_x \mathbf{R}_L C_y$  implies  $C_x \mathbf{R}_{L \cup \{C\}} C_y$ .*

**Proof** If  $C_x \neq C_y$  and  $C_x \mathbf{R}_L C_y$ , then  $C_x$  and  $C_y$  are linked by a sequence  $T_1, T_2, \dots, T_n$  of adaptation triples, each of which includes a case that was uncovered before the addition of  $C$  to the case library. None of these cases can be covered by  $L \cup \{C\}$ , otherwise  $C_y$  would also be covered by  $L \cup \{C\}$ .

However, the two equivalence relations are not the same. Following the addition of a new case to a case library  $L$ , two or more of the equivalence classes defined by  $\mathbf{R}_L$  may merge to form a larger equivalence class under  $\mathbf{R}_{L \cup \{C\}}$ . In general, when a new case  $C$  is added to  $L$ , the equivalence classes (or partitions) defined by  $\mathbf{R}_{L \cup \{C\}}$  are unions of one or more of the equivalence classes defined by  $\mathbf{R}_L$ . To identify the new partitions, it is necessary only to check for adaptation triples that can cause old

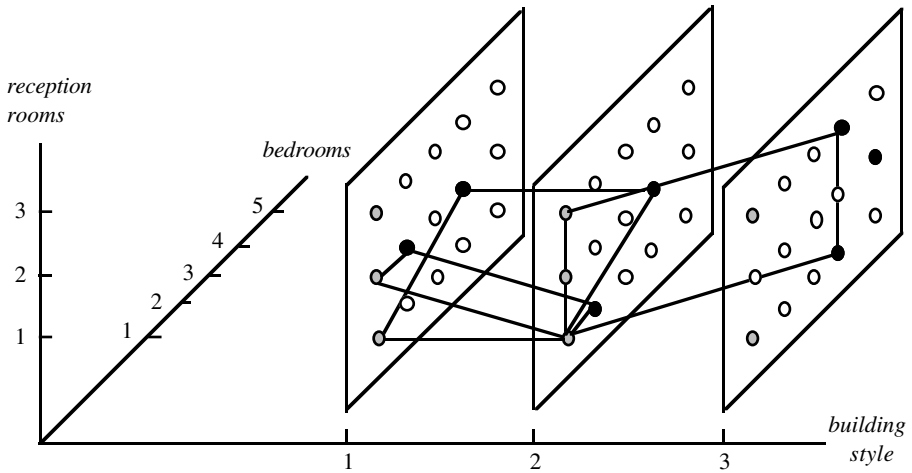


**Fig. 8.** One partition of the uncovered cases in the example problem space

partitions to merge; that is, admissible adaptation triples for still-uncovered, and previously unrelated, cases that include the new library case. This is the strategy used in *rePartition*, the algorithm CaseMaker-2 uses to update the partitions of still-uncovered cases. Again the reversibility of adaptation triples is used to advantage in *rePartition*, which need only examine adaptation triples for the new library case to discover adaptation triples that cause partitions to merge.

## 4 An Example Consultation

We now show CaseMaker-2 being used to guide the selection of cases to increase the coverage provided by the example case library introduced in Section 2, and examine the reasoning on which its recommendations are based. When *disCover* is applied by CaseMaker-2 to the example case library  $L$ , all cases in the  $3 \times 5 \times 3$  problem space are found to be covered except the 18 cases with bedrooms = 1 or 3. One of the 4 partitions of the uncovered cases produced by *eValue* consists of the 8 shaded cases in Fig. 8. For example, three admissible adaptation triples for the uncovered case (2,1,1) are shown in which the non-library case is also an uncovered case. It follows that (2,1,1), (1,1,1), (1,1,2), and (2,1,3) are all in the same  $R_L$  equivalence class. In fact, all cases with bedrooms = 1 belong to the same equivalence class except (3,1,2), which is unrelated to any other case. Similarly, all cases with bedrooms = 3 belong to one  $R_L$  equivalence class except (3,3,2) which is unrelated to any other case. The cases initially recommended by CaseMaker for addition to the case library (Fig. 9) are the 16 cases in the two largest partitions.

The rationale for offering the author as many alternatives as possible is that certain cases may be difficult or impossible for the author to solve. For example, properties with more reception rooms than bedrooms may not be available for

---

CaseMaker-2: Library now contains 7 cases and coverage is 60 per cent (27 cases).  
Any of the following cases will increase coverage to at least 78 per cent (35 cases).

[1,3,1], [2,3,2], [3,3,1], [3,3,3], [2,3,3], [1,3,2], [1,3,3], [2,3,1],  
[1,1,1], [2,1,2], [3,1,1], [3,1,3], [2,1,3], [1,1,2], [1,1,3], [2,1,1]

Which case would you like to add to the library?

User: [2,1,1]

CaseMaker-2: Please enter the solution for [2,1,1]

User: 55

CaseMaker-2: Library now contains 8 cases and coverage is 80 per cent (36 cases).  
Any of the following cases will increase coverage to at least 98 per cent (44 cases).

[1,3,1], [2,3,2], [3,3,1], [3,3,3], [2,3,3], [1,3,2], [1,3,3], [2,3,1]

Which case would you like to add to the library?

User: [1,3,1]

CaseMaker-2: Please enter the solution for [1,3,1]

User: 57

CaseMaker-2: Library now contains 9 cases and coverage is 100 per cent (45 cases).

---

**Fig. 9.** An example case-authoring dialogue in CaseMaker-2

valuation purposes. Following the addition of (2,1,1) and its solution to the case library, reClaim discovers that in addition to the minimum increase in coverage of 8 cases, (3,1,2) can also now be solved. However, no merging of the partitions of the 9 still-uncovered cases is detected by rePartition. The cases now recommended by CaseMaker are therefore the 8 cases in the larger of the two surviving partitions. Following the addition of (1,3,1) to the case library, reClaim discovers that (3,3,2) is also now covered and therefore that full coverage of the target problem space has been achieved.

## 5 Conclusions

CaseMaker-2 is an environment for intelligent case-authoring support in CREST [9-12], incorporating tools for the discovery, application and updating of knowledge to support the authoring process. The discovery of all cases that can be solved from a given case library and evaluation of the coverage contributions of uncovered cases need only be performed once in the life cycle of a case-based reasoner. A *reCommend-reClaim-rePartition* cycle is then repeated in which the coverage contributions of still-uncovered cases are dynamically updated as the case library is extended.

## References

1. Smyth B., Keane M.: Remembering to Forget: a Competence-Preserving Case Deletion Policy for CBR Systems. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montreal (1995) 377-382
2. Zhu J., Yang Q.: Remembering to Add: Competence-Preserving Case-Addition Policies for Case-Base Maintenance. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Stockholm (1999) 234-239
3. Smyth B.: Constructing Competent Case-Based Reasoners: Theories, Tools and Techniques. Proceedings of the IJCAI-99 Workshop on Automating the Construction of Case-Based Reasoners, Stockholm (1999) 17-27
4. Racine K., Yang Q.: Maintaining Unstructured Case Bases. In: Leake D., Plaza E. (eds.): Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence Vol. 1266. Springer-Verlag, Berlin Heidelberg New York (1997) 553-564
5. Aha D., Breslow L.: Refining Conversational Case Libraries. In: Leake D., Plaza E. (eds.): Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence Vol. 1266. Springer-Verlag, Berlin Heidelberg New York (1997) 267-278
6. Bonzano A., Cunningham P., Smyth B.: Using Introspective Learning to Improve Retrieval in CBR : a Case Study in Air Traffic Control. In: Leake D., Plaza E. (eds.): Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence Vol. 1266. Springer-Verlag, Berlin Heidelberg New York (1997) 291-302
7. Leake D., Wilson D.: Categorizing Case-Base Maintenance: Dimensions and Directions. In: Smyth B., Cunningham P. (eds.): Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 196-207
8. Smyth B., McKenna E.: Modelling the Competence of Case-Bases. In: Smyth B., Cunningham P. (eds.): Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 208-220
9. McSherry D.: Relaxing the Similarity Criteria in Demand-Driven Discovery of Adaptation Knowledge. Proceedings of the IJCAI-99 Workshop on Automating the Construction of Case-Based Reasoners, Stockholm (1999) 56-61
10. McSherry D.: An Adaptation Heuristic for Case-Based Estimation. In: Smyth B., Cunningham P. (eds.): Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 184-195
11. McSherry D.: Demand-driven Discovery of Adaptation Knowledge. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Stockholm (1999) 222-227
12. McSherry D.: Automating Case Selection in the Construction of a Case Library. In: Bramer M., Macintosh A., Coenen F. (eds.): Research and Development in Intelligent Systems XVI. BCS Conference Series. Springer-Verlag, London Berlin Heidelberg (1999) 163-177



# Integrating Conversational Case Retrieval with Generative Planning

Héctor Muñoz-Avila<sup>1,2</sup>, David W. Aha<sup>2</sup>, Leonard A. Breslow<sup>2</sup>,  
Dana S. Nau<sup>1</sup>, and Rosina Weber<sup>3</sup>

<sup>1</sup> Department of Computer Science, University of Maryland  
College Park, MD 20742-3255  
{lastname}@cs.umd.edu

<sup>2</sup> Navy Center for Applied Research in AI, Naval Research Laboratory (Code 5510)  
Washington, DC 20375  
{lastname}@aic.nrl.navy.mil

<sup>3</sup> Department of Computer Science, University of Wyoming,  
Laramie, WY 82071

**Abstract.** Some problem-solving tasks are amenable to integrated case retrieval and generative planning techniques. This is certainly true for some decision support tasks, in which a user controls the problem-solving process but cannot provide a complete domain theory. Unfortunately, existing integrations are either non-interactive or require a complete domain theory and/or complete world state to produce acceptable plans, preventing them from being easily used in these situations. We describe a novel integrated algorithm, named SiN, that is interactive and does *not* require a complete domain theory or complete world state. SiN users leverage a conversational case retriever to focus both partial world state acquisition and plan generation. We highlight the benefits of SiN (e.g., quadratically fewer cases needed) in an experimental study using a new travel planning domain.

**Keywords:** Conversational case retrieval, planning, integrations

## 1 Introduction

Mixed-initiative planning is an interactive plan generation process in which two or more independent planners jointly solve a problem. At any time, one of them is controlling the plan generation process. Control is passed either when the planner  $p$  in control *cedes* it to another planner  $p'$ , or through  $p'$ 's *interruption*, which typically occurs when a human planner interrupts a generative planner.

Many systems support mixed-initiative planning, but they are either automated, require a complete domain theory, and/or require a complete world state. This prevents them from being used for (interactive) decision support tasks in which these are unavailable, which characterizes the focal tasks of our current projects (e.g., planning for non-combatant evacuations (NEOs)). We introduce a mixed-initiative plan generation algorithm, named SiN, that integrates a generative planner (SHOP) (Nau et al., 1999) with a conversational case retriever

(NaCoDAE/HTN), an extension of NaCoDAE (Aha & Breslow, 1997), to produce plans in decision support contexts. SiN can be characterized as follows:

- It employs a *unified object representation* (i.e., NaCoDAE/HTN’s cases and SHOP’s methods).
- It uses a *mixed-initiative plan generation process*. In SiN, SHOP cedes control to NaCoDAE/HTN whenever none of SHOP’s methods or operators can be applied, while NaCoDAE/HTN cedes control to SHOP whenever the user completes a conversation.
- It can *generate plans given incomplete domain theories*, which prevents the application of typical generative planners. We will show how SiN’s problem-solving experiences (i.e., NaCoDAE/HTN’s cases) complement generalized domain knowledge (i.e., SHOP’s methods and operators).
- It can *plan with an incomplete world state*. SiN uses NaCoDAE/HTN to interactively perform information gathering activities for SHOP.

After explaining how SiN differs from other planners in Section 2, we detail its hierarchical task network (HTN) representation in Section 3. Section 4 then describes the two modules and details their integration in SiN. We then describe an experimental study in Section 5 to highlight SiN’s benefits.

## 2 Contributions in Comparison with other Planners

The key distinguishing feature of SiN (SHOP interleaved with NaCoDAE/HTN) is its interleaved control structure for plan generation, which is highlighted in Table 1 versus example planning systems in seven categories. SHOP, a generative planner, requires a complete domain theory. CHEF (Hammond, 1989) and DIAL (Leake et. al., 1997) are case-based, but does not exploit a generative component, and thus requires a large case base to perform well across a wide variety of problems. Prodigy/Analogy (Velooso & Carbonell, 1993) integrates generative and case-based planning, but requires a complete domain theory. Similarly, Paris (Bergmann & Wilke, 1995) integrates these two approaches, but is also not interactive.

SIPE II (Wilkins, 1998) is a mixed-initiative generative planner, but, while it can acquire world state information, it requires a complete domain theory. NaCoDAE/HTN (Muñoz-Avila et al., 1999) is a mixed-initiative case-based planner, but, like CHEF, it does not support generative planning. Mitchell’s (1997) architecture, which uses cases to select which task to perform in a given tactical situation, is a more advanced example in this category. Similarly, the CHARADE and CARICA systems (Avesani et al., 1998) interactively acquire state information (i.e., for situation assessment), which is used to retrieve and adapt planning cases that are then given to a resource allocation scheduler. However, they do not support generative planning.

MI-CBP (Velooso et al., 1997), which extends Prodigy/Analogy, uses a control structure where interaction is limited to providing the system with user feedback on completed plans. This requires MI-CBP to input, or learn thru feedback, a

**Table 1.** Contrasting SiN’s characteristics with other planning systems

| System          | Generative | Case-based | Mixed-Initiative | Interleaved |
|-----------------|------------|------------|------------------|-------------|
| SHOP            | ✓          |            |                  |             |
| CHEF            |            | ✓          |                  |             |
| Prodigy/Analogy | ✓          | ✓          |                  |             |
| SIPE II         | ✓          |            | ✓                |             |
| NaCoDAE/HTN     |            | ✓          | ✓                |             |
| MI-CBP          | ✓          | ✓          | ✓                |             |
| SiN             | ✓          | ✓          | ✓                | ✓           |

sufficiently complete domain theory to solve problems. In contrast, SiN gathers information it requires from the user through NaCoDAE/HTN conversations, but does not learn from user feedback. CAPlan/CbC (Muñoz-Avila et al., 1997) is another integrated, interactive planner, but its interaction does not include acquiring world state information.

Finally, SiN’s interleaved control strategy allows both the case-based and generative planning modules to contribute task decompositions during planning. Because the SiN user supplies world state information incrementally thru its interaction with NaCoDAE/HTN, as needed, not all of the world state is needed a priori to generate plans.

Perhaps the most closely related architecture to SiN is the one described by Carrick et al. (1999), which uses pre-stored hierarchical plans to perform information gathering activities for a conversational case retriever to solve interactive diagnosis tasks. SiN instead uses a conversational case retriever to gather information, and provide task decompositions, for a generative planner. Integrating SiN with their approach would yield a powerful interactive planner, especially if we do not require the information-gathering plans to be pre-constructed. This is an interesting avenue for future work.

SiN is a subset of an extended HICAP (Muñoz-Avila et al., 1999). We summarize their relationship in Section 6. Figure 1 shows a snapshot of HICAP. It displays a plan for trip from Greenbelt to downtown New York City (NYC). The left side shows a hierarchy of *tasks* and the right side a hierarchy of *resources*. For the rest of this paper we will concentrate on the generation of task hierarchies.

### 3 Hierarchical Task Network Plans

This paper concerns a mixed-initiative elicitation process for generating task hierarchies. A *task hierarchy*, is a triple  $(T, <, \wedge)$ , where  $T$  is a set of *tasks*. The relation  $<$  is an *ordering relation* between elements in  $T$ . We write  $t1 < t2$  if  $t2$  should be executed after  $t1$ . In Figure 1, the  $<$  relations are represented by the arrows. For example, the arrow from the task *Travel-segment from Greenbelt to Greenbelt-Metro* to the task *Travel-segment from Greenbelt-Metro to Union Station* indicates that *Travel-segment from Greenbelt to Greenbelt-Metro*  $<$  *Travel-*

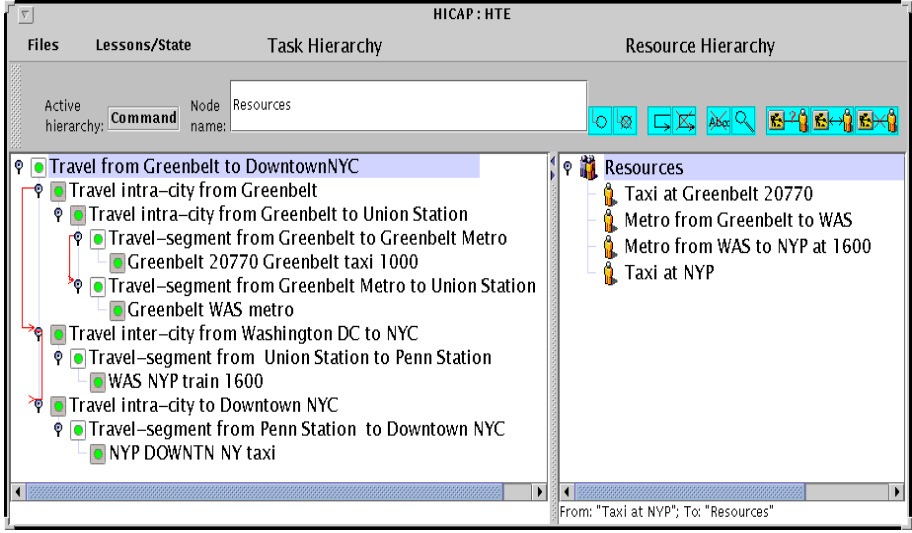


**Fig. 1.** A snapshot of HICAP

*segment from Greenbelt-Metro to Union Station* holds. The notation  $\wedge$  denotes a hierarchical relation, where  $t1 \wedge t2$  indicates that  $t2$  is a subtask of  $t1$ . Hierarchical task relations are represented by indentations in Figure 1. For example, the task *Travel inter-city from Washington DC to NYC* is a subtask of *Travel from Greenbelt to Downtown NYC*. The task hierarchy is displayed as a tree (see Figure 1) with a root indicating the top-level task, called the *initial task*. Tasks that are not decomposable are called *primitive*. The others are called *compound*.

Given a task hierarchy, The ordering ( $<$ ) and hierarchical ( $\wedge$ ) relations induce a hierarchical ordering relation ( $<_{\wedge}$ ) as follows: given two tasks  $t1$  and  $t2$ , the ordering relation  $t1 <_{\wedge} t2$  holds iff:

- $t1 < t2$  holds, or
- there are ancestors  $t1'$  of  $t1$  in  $T$  and  $t2'$  of  $t2$  in  $T$ , w.r.t.  $\wedge$ , such that  $t1' < t2'$ ,  $t1 < t2'$ , or  $t1' < t2$  holds.

In Figure 1, *Travel intra-city from Greenbelt*  $<_{\wedge}$  *Travel-segment from Union Station to Penn Station* holds because *Travel inter-city from Washington DC to NYC* is an ancestor of *Travel-segment from Union Station to Penn Station* and *Travel intra-city from Greenbelt*  $<$  *Travel inter-city from Washington DC to NYC* holds.

Given a task hierarchy  $(T, <, \wedge)$ , its *concrete plan*, or simply *plan*, is a pair  $(T_{action}, <_{\wedge})$ , where  $T_{action}$  is the set of all primitive tasks in  $T$ . For example, the concrete plan associated with the task hierarchy shown in Figure 1 consists of four steps:

1. travel by taxi from Greenbelt to the Greenbelt Metro Station,
2. take the Metro to Union Station,
3. take the train to Penn Station in NYC, and
4. take a taxi from Penn Station to downtown NYC.

Hierarchical task network (HTN) planning proceeds by decomposing a given task into simpler tasks. A *domain theory* indicates how tasks are to be decomposed; we will detail this in Section 4. A *state* is a set of ground atoms that denote known true facts. A *planning problem* is a triple  $(t, S, D)$ , where  $t$  is as task,  $S$  is a state and  $D$  is a domain theory.  $t$  becomes the initial task of the resulting task hierarchy.

## 4 Integrating Case Retrieval with Plan Generation

This section introduces a novel approach for interleaving case retrieval and plan generation in a decision support context. The first two subsections describe the individual modules, while the third describes their integration.

### 4.1 SHOP: A Simple Hierarchical Ordered Planner

SHOP (Nau et al., 1999) is a domain-independent HTN planner that plans for tasks in the order that they will be executed. SHOP has been shown to be several orders of magnitude faster than some well-known AI planning systems.

SHOP applies a domain theory, consisting of methods and operators, to generate plans. A *method* is an expression of the form  $M = (h, P, ST)$ , where  $h$ , (the method's *head*) is a compound task,  $P$  is a set of *preconditions*, and  $ST$  is the set of  $M$ 's (children) *subtasks*.  $M$  is *applicable* to a task  $t$ , relative to a state  $S$ , iff  $\text{matches}(h, t, S)$  (i.e.,  $h$  and  $t$  have the same predicate and arity, and a consistent set of bindings  $\Theta$  exists that maps variables to values such that all terms in  $h$  match their corresponding ground terms in  $t$ ) and the preconditions  $P$  are *satisfied* in  $S$  (i.e., there exists a consistent extension of  $\Theta$ , named  $\Theta'$ , such that  $\forall p \in P \{p\Theta' \in S\}$ ), in which case  $M(t, S) = ST\Theta'$ .

Figure 2(a) displays an example method that defines how to move between two locations. The preconditions for applying this method indicate that the cities in these locations must have train stations. The subtasks are to travel to the train station, travel between the two train stations, and then travel to the final location. (SHOP always assume a total order between the subtasks; the subtasks are planned in the order that they are listed in the method.) This was the method applied to the compound task *travel from Greenbelt to Downtown NYC* in Figure 1. This task is represented internally as *travel(Greenbelt, DowntownNYC)*. The variable bindings for the parent task of this method were  $\{ ?locFrom \rightarrow \text{Greenbelt}, ?locTo \rightarrow \text{DowntownNYC} \}$ .

An *operator* is an expression of the form  $O = (h, A, D)$ , where  $h$  (the operator's *head*) is a primitive task, and  $A$  and  $D$  are the so-called *add-* and *delete-*lists. These lists define how the operator's application transforms the current state  $S$ :

|                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Head:</b><br/> <i>travel(?locFrom, ?locTo)</i></p> <p><b>Preconditions:</b><br/> <i>in(?locFrom, ?city1)</i><br/> <i>in(?locTo, ?city2)</i><br/> <i>trainStation(?station1, ?city1)</i><br/> <i>trainStation(?station2, ?city2)</i></p> <p><b>Subtasks:</b><br/> <i>travelC(?locFrom, ?station1)</i><br/> <i>travelIC(?station1, ?station2)</i><br/> <i>travelC(?station2, ?locTo)</i></p> | <p><b>Head:</b><br/> <i>travelC(PennSt, DowntownNYC)</i></p> <p><b>Questions:</b><br/>         Is it raining hard in NYC? : No<br/>         Is the traveler carrying luggage? : Yes</p> <p><b>Subtasks:</b><br/> <i>take(taxi, PennSt, DowntownNYC)</i></p> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Fig. 2.** (a) Example of a method in the personal travel domain. Terms beginning with a question mark denote variables. (b) Case from the travel domain

every element in the add-list is added to  $S$  and every element in the delete-list is removed from  $S$ . An operator  $O$  is *applicable* to a task  $t$ , relative to a state  $S$ , iff matches( $h, t, S$ ).

*SHOP Planning Algorithm.* At any point during the planning process, SHOP is decomposing a task hierarchy  $(T, <, \wedge)$  relative to a state  $S$ . The initial task is the focus of the current problem  $(t, S, D)$ . Leaves in the task hierarchy are either decomposable or not decomposable (i.e., SHOP has no decomposition methods or applicable operators for them). SHOP plans, using preorder traversal on the leaf tasks, as follows:

1. If it finds a leaf task  $t$  that is primitive and has an applicable operator  $O$ , then  $O$  is applied to  $t$ , and  $S$  is updated as needed.
2. Else if  $t$  is compound and an applicable method  $M$  exists, then  $M$  is applied, which expands  $t$  with  $M$ 's subtasks.
3. Else if all of the leaves in the task hierarchy are primitive, this process terminates with a success.
4. Else, SHOP backtracks.

## 4.2 NaCoDAE/HTN: A Conversational Case Retriever

NaCoDAE/HTN is an extension of the NaCoDAE conversational case retriever (Aha & Breslow, 1997) that can be used to suggest task decompositions. A *case* is an expression of the form  $C = (h, QA, ST)$ , where  $h$  is the case's head (i.e., a task),  $QA$  is a list of ⟨question, answer⟩ pairs, and  $ST$  is a set of subtasks.

Users interact with NaCoDAE/HTN in *conversations*, which begin when the user selects a task  $t$ . NaCoDAE/HTN responds by displaying the top-ranked

cases whose heads are equal to  $t$ . Cases are ranked according to their *similarity* to the current state  $S$ , which is the state that exists at that time during the conversation. Similarity is computed by comparing the contents of the current state with a case's question-answer pairs. (That is, each question-answer pair is represented as a monadic predicate in  $S$ , and similarity for a given  $\langle \text{question}, \text{answer} \rangle$  pair becomes a membership test in  $S$ .) NaCoDAE/HTN also displays questions, whose answers are not known in  $S$ , ranked according to their frequency among the top-ranked cases. The user can select and answer (with  $a$ ) any displayed question  $q$ , which inserts  $q(a)$  into  $S$ . This state change subsequently modifies the case and question rankings. A conversation ends when the user selects a case  $C$ , at which time  $t$  is decomposed into  $ST$  (i.e.,  $C$ 's subtasks).

Figure 2(b) displays an example case from the personal travel domain. A person traveled by taxi between Penn Station and downtown because it was not raining and the traveler has luggage. In Figure 1, this was the case applied to decompose the task *Travel-segment from Penn Station to Downtown NYC* (internally represented as *travelC(PennSt,DowntownNYC)*) into *NYP DOWNTN NY taxi* (internally represented as *take(taxi,PennSt,DowntownNYC)*).

*NaCoDAE/HTN Case-Based Planning Algorithm.* As with the SHOP generative planner, NaCoDAE/HTN refines task hierarchies. When given a task  $t$  to refine, NaCoDAE/HTN uses  $t$  as an index for initial case retrieval and conducts an interactive conversation, which ends when the user selects a case  $C$ , which is then used to decompose  $t$  into  $ST$  (i.e.,  $C$ 's subtasks).

### 4.3 The SiN Integration

SHOP's methods and NaCoDAE's HTN cases are similar in that they indicate how to decompose a task into subtasks. Their main difference is that a method's preconditions must always be satisfied in the current state, which need not be true for a case's  $\langle q, a \rangle$  pairs (i.e., cases support partial matching). Also, a user can select and apply a case's task decomposition even when some of its questions have not been answered. The decision to apply a case is the user's responsibility.

This difference can be illustrated with the example method and case shown respectively in Figures 2(a) and 2(b). In the method, one can only use a train if there is a train station in both locations. However, the case can be applied (i.e., use the taxi) even if it is sunny or the traveler is not carrying substantial luggage.

A single (current) state  $S$  is maintained in SiN that is accessible to and updatable by both SHOP and NaCoDAE/HTN. Answers given by the user during an interaction with NaCoDAE/HTN are added to  $S$  (i.e., each question has a translation into a first-order predicate). Changes that occur by applying SHOP's operators are also reflected in  $S$ .

Both SHOP and NaCoDAE/HTN assist SiN with refining task hierarchies. At any point of time, one of these is in control and is focusing on a compound, leaf task  $t$  to decompose. SiN proceeds as follows:

1. If SHOP is in control and can decompose  $t$ , it does so. If SHOP cannot decompose  $t$ , and if  $t$  can be decomposed by NaCoDAE/HTN, SHOP cedes control to NaCoDAE/HTN.
2. If NaCoDAE/HTN is in control and can decompose  $t$ , it does so. If NaCoDAE/HTN cannot decompose  $t$ , and  $t$  can be decomposed by SHOP, NaCoDAE/HTN cedes control to SHOP.
3. If neither SHOP nor NaCoDAE/HTN can decompose  $t$ , SHOP will backtrack, if possible. If backtracking is impossible (for example, because  $t$  is the initial task), this planning process is interrupted and a failure is returned.

By continuing in this way, and assuming that the process is not interrupted with a failure, SiN will eventually yield a task hierarchy in which all leaves are primitive tasks. Typically, this final task hierarchy contains decompositions generated by SHOP and others generated by NaCoDAE/HTN. The task hierarchy depicted in Figure 1 illustrates this situation; all interior nodes (i.e., nodes that are neither the root nor leaves) are generated by SHOP whereas all leaves are generated by NaCoDAE/HTN.

## 5 Evaluation

As explained in Section 2, planning algorithms comparable to SiN require a complete domain theory, a complete world state, or, as explained in Section 5.2, a much larger knowledge base. Therefore, empirical comparisons with these other planning algorithms would not be particularly revealing; SiN is in a different category of planners.

Therefore, for this evaluation, we focused on our claim that SiN’s capabilities extend those of its component algorithms. Our arguments rest on brief theoretical and empirical analyses. Section 5.1 focuses on the simpler argument: comparing SiN with SHOP. Section 5.2 then compares SiN with NaCoDAE/HTN.

### 5.1 Comparing SiN with SHOP

Because SHOP is a generative planner, it requires a complete world state and domain theory to solve a given planning problem. Otherwise, if one of its methods requires world state information for testing its preconditions, and that information is unavailable, then it cannot be applied, and SHOP will fail to generate a plan. Furthermore, if it is not given a complete domain theory, then it may again fail when it is unable to generate a decomposition for a given task.

In contrast, SiN does not require a complete world state and domain theory; it uses NaCoDAE/HTN to incrementally collect world state information, and to provide decomposition suggestions for tasks that SHOP cannot decompose (i.e., because it lacks methods for those tasks). In regard to the world state, SiN uses NaCoDAE/HTN to elicit, from the user, only the world state information it needs to generate a certain plan. In comparison, because the user doesn’t know in advance what information is necessary for SHOP to infer a plan, they must encode more information about the world state that is required by SHOP to generate a plan.



**Table 2.** Questions used to define personal travel states

| Question                                            | Possible Answers    |
|-----------------------------------------------------|---------------------|
| Is it raining hard?                                 | Y/N                 |
| Is there any major accident or unusual event?       | Y/N                 |
| What is the chance of large snow accumulation?      | High, moderate, low |
| How many passengers?                                | 1, 2, 3, 3+         |
| Is there any luggage to check?                      | Y/N                 |
| Are any children or seniors involved?               | Y/N                 |
| Is travel occurring on a holiday? Which?            | Holiday name, or No |
| What is the starting day of the week for this trip? | 7 distinct answers  |

## 5.2 Comparing SiN with NaCoDAE/HTN

SiN’s advantage vs. using only NaCoDAE/HTN is that, for domains requiring different plans for different world states, NaCoDAE/HTN will need significantly more cases to generate the same space of plans as SiN. It is important for NaCoDAE/HTN to generate the same space of plans as SiN because plan quality varies greatly with the selected dependent measure(s). We first provide empirical evidence for this claim, and then focus on a theoretical justification.

The purpose of our experimental study was to investigate how the quality of the generated plans vary. Towards this goal, we developed a knowledge base and plan evaluator for a planning domain where the contents of the world state can significantly impact choices during planning. Named the *personal travel domain*, its plans concern traveling from locations in Washington, DC to downtown New York City (NYC). By encoding 7 transportation methods (3 inter- and 4 intra-city) and defining plans to have 3-5 segments, approximately 756 plans could be generated. We defined states using the eight questions shown in Table 2. The knowledge base consists of 10 methods and 1 operator for SHOP and 40 cases for NaCoDAE/HTN. Each plan’s task hierarchy was obtained by applying 4-6 methods, 3-5 operators (i.e., the same one repeatedly), and 3-5 cases.

Our personal travel plan evaluator can generate a different time duration each time it is given a plan and world state because of its nondeterministic execution. For each run, it outputs whether the plan succeeded and, if so, the trip’s duration. A plan fails when segment delays cause a late arrival for a segment requiring a fixed time departure (e.g., an airplane flight). For each segment, we applied a delay function that is influenced by world state conditions. For example, a flight segment will incur a longer delay for higher chances of large snow accumulation, especially on holidays (i.e., high travel days). Segments are categorized into short, medium, and long lengths, and delays can range from 0 up to 4.5 times a segment’s anticipated duration; smaller multiples are used for maximum delays for medium (3.5) and long (2.5) duration segments.

We selected ten goals, corresponding to ten pairs, each consisting of a departure location in Washington, DC and downtown NYC. For each goal, we generated 10 random world states (i.e., providing answers to all 8 questions), thus yielding 100 total planning problems. SiN was then used to generate a plan

**Table 3.** Comparative results on the travel domain

| Algorithm                        | Success Rate | Mean Duration      |
|----------------------------------|--------------|--------------------|
| 100 plans                        | 70%          | 9h45min (3h14min)  |
| Plans with highest success rates | 92%          | 13h35min (3h13min) |
| Plans with lowest success rates  | 45%          | 7h40 (3h 7min)     |

for each problem, and each was executed 10 times by the plan evaluator. The first row of Table 3 displays the results.

We examined the output of the evaluator obtained for the 100 plans generated. For each goal  $g$ , we selected two plans; one for which the success rate was the highest and another for which the success rate was the lowest. For each of the 10 goals, we selected the plans with highest and lowest success rates and re-ran the evaluator, again using the 10 random world states used previously. The results are displayed in rows 2 and 3 of Table 3, along with parenthesized standard deviations. We observed a correlation between the success rate and the duration. For the plans with highest success rate, the duration increases by almost 4 hours, or 39% over the mean computed over the 100 plans. For the plans with the lowest success rate, the duration decreases by approximately 2 hours, or 22% relative to the mean.

These results show that there is a large variation in the quality of the plans that potentially could be generated by SiN in terms of the success rate and the duration. Because it is an interactive algorithm, SiN can generate plans with widely different qualities according to an *individual* user's needs by using NaCoDAE/HTN. Some users may want to infer plans that they think will be successful, others may want to infer plans with lower duration, and yet others may want plans with both lower duration and higher success rate. There are other measures of quality, including cost and waiting time (i.e., how long the traveler has to wait in the airport). We conclude from these experiments that, to provide a similar flexibility as in SiN, NaCoDAE/HTN must generate plans from the same space.

We can approximate the number of cases that NaCoDAE/HTN, working solo, would require to generate plans from the same space as SiN does by carefully analyzing each of SHOP's methods for this domain. These methods are used for all *interior* branches in the task hierarchy generated by SiN; cases, and the sole SHOP operator, are used for task decompositions only at the *bottom* branches (i.e., leading to leaves). Using the personal travel domain, Table 4 shows the number of methods and cases required by SiN and NaCoDAE/HTN, based on three domain characteristics. In summary, the number of methods required by SiN's SHOP module is constant, while the number of cases required by NaCoDAE/HTN to cover the same hierarchical decompositions as SHOP's methods increases quadratically with domain size. For example, with the current values of  $L_s = 10$  and  $L_d = 1$ , the number of cases required by NaCoDAE/HTN to encode SHOP's operators is 165. But this rises to 1248 when  $L_s = 31$  and  $L_d = 4$

**Table 4.** The number of SHOP methods or NaCoDAE/HTN cases required for the personal travel domain, where  $L_s$  is the number of starting locations,  $L_d$  is the number of destination locations, and  $N$  is the number of possible plan segments

| Branch Type     | SiN  |             | NaCoDAE/HTN         |
|-----------------|------|-------------|---------------------|
|                 | SHOP | NaCoDAE/HTN |                     |
| <b>Interior</b> | 10   | 0           | $O(L_s(L_s + L_d))$ |
| <b>Bottom</b>   | 0    | $N$         | $N$                 |

(i.e., the number of start and destination locations that are known by the plan evaluator).

In summary, SiN effectively combines abstract and experiential knowledge to plan, where the abstract knowledge (SHOP’s methods) refers to generalizations in planning space. This is a far more compact representation than using only experiential knowledge (i.e., cases), and is reminiscent of (non-planning) systems that effectively combine rules and cases (e.g., Golding & Rosenbloom, 1991).

## 6 Summary and Relation to HICAP

In this paper we introduced SiN, a mixed-initiative planner that interleaves case-based and generative planning modules. We argued that this structure provides SiN with benefits relative to its individual components; it relaxes its generative planner’s need for a complete world state and domain theory, and it can significantly reduce the number of cases required by its conversational case retriever. Furthermore, SiN is applicable to decision support tasks, and we summarized its benefits vs. other integrated, case-based planners.

SiN is a compound module of HICAP, the plan authoring tool that previously did not contain a generative planning module (Muñoz-Avila et al., 1999). In addition to SiN’s two modules, HICAP’s other modules include the HTE interface, the DecTS conflict resolution manager, and an active delivery module for lessons learned (Weber et al., 2000). Future work on HICAP includes incorporating a learning apprentice for assisting with knowledge (i.e., cases, methods, lessons) acquisition, incorporating case-adaptation capabilities, and integrating it with a more sophisticated front-end (e.g., SOFTools) for incremental planning.

## Acknowledgments

This research was supported by grants from the Office of Naval Research, the Naval Research Laboratory, and the Army Research Laboratory. Thanks also to our submission’s reviewers.

## References

1. Aha, D. W., & Breslow, L. A. (1997). Refining conversational case libraries. *Proceedings of the Second International Conference on CBR* (pp. 267–278). Providence, RI: Springer.
2. Avesani, P., Perini, A., & Ricci, F. (1998). The twofold integration of CBR in decision support systems. In D. W. Aha & J. J. Daniels (Eds.), *Case-Based Reasoning Integrations: Papers from the 1998 Workshop* (Technical Report WS-98-15). Menlo Park, CA: AAAI Press.
3. Bergmann R., & Wilke W. (1995). Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, 3, 53–118.
4. Carrick, C., Yang, Q., Abi-Zeid, I., & Lamontagne, L. (1999). Activating CBR systems through autonomous information gathering. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 74–88). Munich, Germany: Springer.
5. Golding, A., & Rosenbloom, P. S. (1991). Improving rule-based systems through case-based reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 22–27). Anaheim, CA: AAAI Press.
6. Hammond, K. (1989). *Case-based planning: Viewing planning as a memory task*. Boston, MA: Academic Press.
7. Leake, D., Kinley, A., & Wilson, D. (1997). A Case Study of Case-Based CBR. *Proceedings of the Second International Conference on Case-Based Reasoning* (pp. 371–382). Providence, RI: Springer.
8. Mitchell, S. W. (1997). A hybrid architecture for real-time mixed-initiative planning and control. *Proceedings of the Ninth Conference on Innovative Applications of AI* (pp. 1032–1037). Providence, RI: AAAI Press.
9. Muñoz-Avila, H., McFarlane, D., Aha, D. W., Ballas, J., Breslow, L. A., & Nau, D. (1999). *Using guidelines to constrain interactive case-based HTN planning*. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 288–302). Munich, Germany: Springer.
10. Muñoz-Avila, H., Weberskirch, F., & Roth-Berghofer, T. (1997). On the relation between the context of a feature and the domain theory in case-based planning. *Proceedings of the Second International Conference on Case-Based Reasoning* (pp. 337–348). Providence, RI: Springer.
11. Nau, D., Cao, Y., Lotem, A., & Muñoz-Avila, H. (1999). SHOP: Simple hierarchical ordered planner. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. 968–973). Stockholm: AAAI Press.
12. Veloso, M. M., & Carbonell, J. G. (1993). Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, 10, 249–278.
13. Veloso, M., Mulvehill, A. M., & Cox, M. T. (1997). Rationale-supported mixed-initiative case-based planning. *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence* (pp. 1072–1077). Providence, RI: AAAI Press.
14. Weber, R., Aha, D. W., Muñoz-Avila, H., & Breslow, L. A. (2000). Active delivery for lessons learned systems. To appear in *Fifth European Workshop on Case-Based Reasoning*. Trento, Italy: Springer.
15. Wilkins, D. E. (1998). *Using the SIPE-2 planning system: A manual for Version 5.0* (Working Document). Menlo Park, CA: Stanford Research International, Artificial Intelligence Center.

# A Symmetric Nearest Neighbor Learning Rule

Richard Nock<sup>1</sup>, Marc Sebban<sup>1</sup>, and Pascal Jappy<sup>2</sup>

<sup>1</sup> Université des Antilles-Guyane, TRIVIA-Dept of Maths and CS  
97159 Pointe-à-Pitre, France  
{[rnock](mailto:rnock@univ-ag.fr),[msebban](mailto:msebban@univ-ag.fr)}@univ-ag.fr

<sup>2</sup> Université des Antilles-Guyane, Lab. of Tropical Atmosphere Physics  
97159 Pointe-à-Pitre, France  
[dbernard@univ-ag.fr](mailto:dbernard@univ-ag.fr)

**Abstract.** In this paper, we propose a thorough investigation of a nearest neighbor rule which we call the “Symmetric Nearest Neighbor (sNN) rule”. Basically, it symmetrises the classical nearest neighbor relationship from which are computed the points voting for some instance. Experiments on 29 datasets, most of which are readily available, show that the method significantly outperforms the traditional Nearest Neighbors methods. Experiments on a domain of interest related to tropical pollution normalization also show the greater potential of this method. We finally discuss the reasons for the rule’s efficiency, provide methods for speeding-up the classification time, and derive from the sNN rule a reliable and fast algorithm to fix the parameter  $k$  in the  $k$ -NN rule, a longstanding problem in this field.

## 1 Introduction

The Nearest Neighbor (NN) rule is one of the simplest and oldest classification techniques related to Case-Based Reasoning (CBR). It uses a set of observations, also called reference sample,  $\mathcal{S}$ , from a  $n$ -dimensional metric space  $\mathcal{X}$  to classify members of  $\mathcal{X}$  in one of  $c$  classes based on the classification of the elements of  $\mathcal{S}$ . For each  $x \in \mathcal{X}$ , we choose the element  $x' \in \mathcal{S}$  that is nearest to  $x$  (with respect to a given metric), and assign to  $x$  the same class as  $x'$ . In case of ties, a tie breaking rule chooses randomly for  $x'$  one of the nearest neighbors. This rule is obviously encompassed by the following one: the nearest neighbor of  $x$  can be replaced by a voting set of  $k$  of the nearest instances of  $x$ , the  $k$ -nearest neighbors ( $k$ -NN), a majority vote giving the class of  $x$ . Numerous experiments record that simple nearest neighbor methods are competitive with very sophisticated classifiers on challenging problems [9]. Since the first appearance of a  $k$ -NN-like rule [6], domains such as memory-based, instance-based, local, or lazy learning have contributed to its widespread use, and much work was done in and around the NN paradigm to improve the rule. In particular, recent publications [13,14] have focused in locally adapted neighborhoods, transforming the spherical neighborhoods centered on some instance  $x$  into ellipsoidal neighborhoods, elongated to the directions of constant posterior probability, and narrowed in the directions

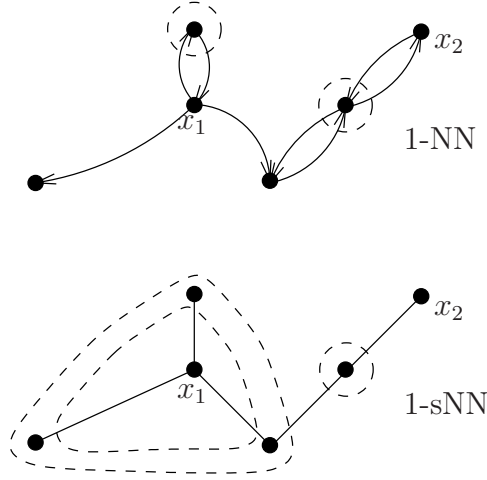


**Fig. 1.** A simple example showing neighborhoods of the 1-NN and 1-sNN rules for  $x_1$  and  $x_2$

of greatest variations. Their objective was to reduce the curse of dimensionality which states that, as dimensionality increases, the average or median neighborhood size on the uniform distribution shrinks for the *nearest neighbor* only as  $|\mathcal{S}|^{1/n}$ , thereby leading to large neighborhoods and possibly less reliable classification. We are going to show that a very simple algorithm of low complexity, also locally adapting neighborhoods, improves the classification of the nearest neighbor rule on most of 29 benchmark problems.

In the “Symmetric Nearest Neighbor rule”,  $k$ -sNN, we make vote for some  $x$  the points in  $\mathcal{S}$  which could belong to the  $k$  nearest neighbors of  $x$ , and the points in  $\mathcal{S}$  for which  $x$  could belong to the  $k$  nearest neighbors. For example, if some  $x$  could vote for some  $y$  in the  $k$ -NN rule, then  $y$  votes for  $x$  in the  $k$ -sNN rule, even if  $y$  does not vote for  $x$  in the  $k$ -NN rule. Figure 1 shows a simple case with 6 points in the plane and the neighborhoods corresponding to the 1-NN and 1-sNN rules for two points  $x_1$  and  $x_2$ . Arcs  $(x, y)$  in the first part indicates that  $x$  can vote for  $y$  in the 1-NN rule. In the 1-sNN rule, arcs are replaced by edges, and the neighborhood of  $x_1$  contains three points instead of one in the 1-NN rule. Our motivation to address this approach and compare it to the traditional  $k$ -NN rule is simple: in the set  $\mathcal{S} \cup \{x\}$ , the  $k$ -sNN rule increases the number of voting neighbors compared to the  $k$ -NN rule, but *without* actually creating “new” votes: for any  $y$  voting for  $x$  in the  $k$ -sNN rule, either  $y$  could vote for  $x$  or  $x$  could vote for  $y$  in the  $k$ -NN rule. This is a way to increase the number of voters while still working on the neighborhood graph of the  $k$ -NN rule, thus keeping the same maximal neighborhood size, with possibly more reliable classification at least for reasonable values of  $k$ . The motivation of this paper is precisely to test this assertion.

These additional neighbors, which we also call *adjuncts*, were previously used in Machine Learning (ML) to remove irrelevant instances from  $\mathcal{S}$ , but not in subsequent classification phases [19]. Also, the graph of *symmetric neighborhoods* was previously used in the field of image processing, but in that case for filter design, and without link to ML purposes [12].

Compared to [13,14], we think of our approach as providing a conceptually simple solution at low computational cost, for which experiments tend to show that it does not trade accuracy for simplicity. To demonstrate this fact, the remainder of the paper is mostly experimental. First, we present results on 29 datasets, most of which come from the UCI repository of ML database [2]. These are complemented by tests on a problem related to pollution normalization in Tropical Physics for which ML represents a solution of great potentiality (yet no real investigation in official texts has been proposed to date). In the last part, we extend the discussion of the sNN rule’s reliability, we present methods to speed-up the classification of new instances, and provide tests on a particular longstanding problem to which the sNN neighborhood graph brings an accurate solution at affordable complexity: the choice of  $k$  for the  $k$ -NN rule.

## 2 Experimental Results on the sNN Rule

### 2.1 The 1-sNN Rule Beats NN Rules

In this Section, we compare the 1-NN rule with its equivalent tNN and sNN rules (the  $k$ -tNN rule where “t” stands for total, is the  $k$ -NN rule with the slight difference that when there are ties, *all* points at distance not exceeding that of the  $k^{th}$  nearest neighbor vote). Table 1 presents the results that were obtained. Four accuracies are given for each dataset, computed using Leave-One Out Cross Validation (LOOCV). The first third are the accuracies of the 1-NN, 1-tNN and 1-sNN rules respectively, and the fourth is the accuracy of the  $k$ -NN classifier with  $k$  equal to the average neighborhood size (without the instance to be classified) of the 1-sNN classifier. This rule is denoted  $\bar{k}_{sNN}$ -NN; it is used to compare the 1-sNN with a conventional NN rule with a similar number of neighbors (on average). The last columns display the average neighborhood sizes for 1-tNN and 1-sNN over all runs and all instances (remark that this represents the average degree of the neighborhood graphs over  $\mathcal{S}$ ). Two major factors can improve the performances of the 1-sNN rule compared to the 1-NN rule’s. The first is the increasing number of voting instances, the second is the better “construction” (or choice) of the neighbors. The fourth rule was run in order to reduce the former effect, and evaluate the latter, much more interesting to us. The results of the 1-tNN rule displayed by Table 1 show that the simple increase of the number of neighbors compared to the 1-NN rule tends indeed to favor greater accuracies, yet the difference is not significant: a sign test cannot separate the two first columns even at risk  $\alpha \leq 30\%$ . Moreover, when removing the atypical Monks 1 problem for which the 1-NN rule performs poorly, 1-tNN beats the 1-NN rule only by 0.4% on average. The results of the 1-sNN rule are much more striking. 1-sNN beats almost systematically the 1-NN and 1-tNN rules together. One

**Table 1.** Accuracies on LOOCV for five rules. Average resulting neighborhood’s sizes are indicated for 1-tNN and 1-sNN. When accuracies of either 1-NN, 1-tNN or 1-sNN are greater than or equal to each of the *four* accuracies, the results are bold faced (these represent very good results on the corresponding problems)

| Dataset       | Accuracy     |              |              |                    | $\bar{k}$ |       |
|---------------|--------------|--------------|--------------|--------------------|-----------|-------|
|               | 1-NN         | 1-tNN        | 1-sNN        | $\bar{k}_{sNN-NN}$ | 1-tNN     | 1-sNN |
| Audiology     | 78.30        | 78.30        | <b>78.80</b> | 74.80              | 1.83      | 2.85  |
| Australian    | 77.70        | 77.50        | 78.00        | 80.10              | 1.77      | 2.62  |
| Balance       | 79.20        | 80.30        | 80.50        | 81.00              | 6.40      | 6.80  |
| Bigpole       | 65.00        | 64.40        | <b>65.40</b> | 62.80              | 3.75      | 4.64  |
| Breast-W      | 95.60        | 96.30        | 96.40        | 96.90              | 1.66      | 7.42  |
| Car           | 89.20        | 88.40        | <b>99.30</b> | 97.70              | 2.00      | 7.28  |
| Crx           | 79.00        | 79.00        | <b>80.40</b> | 78.40              | 1.94      | 2.93  |
| Dermatology   | 90.00        | 89.30        | <b>90.40</b> | 90.00              | 1.12      | 1.69  |
| Echo          | 54.20        | 54.70        | <b>63.40</b> | 58.80              | 2.40      | 3.42  |
| German        | 68.10        | 71.70        | <b>72.50</b> | 72.30              | 3.23      | 4.94  |
| Glass2        | 66.30        | 66.90        | <b>74.20</b> | 73.60              | 6.03      | 7.37  |
| Hard          | <b>58.20</b> | 53.10        | 48.80        | 50.40              | 2.82      | 3.76  |
| Heart         | 75.00        | 75.00        | 75.10        | 76.66              | 1.86      | 2.73  |
| Hepatitis     | 83.20        | 84.50        | <b>86.50</b> | 81.30              | 1.84      | 2.90  |
| Horse Colic   | 71.20        | <b>71.70</b> | 70.90        | 70.90              | 1.18      | 1.95  |
| Ionosphere    | 84.30        | 83.20        | <b>84.90</b> | 78.90              | 4.16      | 6.41  |
| LED Even      | 87.40        | 87.20        | 87.60        | 88.80              | 18.77     | 19.73 |
| LED Even+17   | 65.50        | 68.50        | 72.00        | 75.50              | 1.86      | 2.86  |
| Lung Cancer   | 53.10        | 50.00        | <b>62.50</b> | 53.10              | 1.09      | 1.81  |
| Monks 1       | 66.40        | 98.60        | <b>99.80</b> | 97.00              | 7.50      | 7.80  |
| Pima          | 67.10        | 68.90        | 69.80        | 70.40              | 9.32      | 10.73 |
| Tic tac toe   | 75.00        | 76.00        | <b>84.00</b> | 82.00              | 1.81      | 6.32  |
| Vehicle       | 65.00        | 71.30        | <b>72.00</b> | 71.00              | 6.35      | 7.53  |
| Waves         | <b>76.80</b> | 75.60        | 75.20        | 76.80              | 1.44      | 2.34  |
| WhiteHouse    | 92.40        | 92.00        | <b>92.90</b> | 91.30              | 3.23      | 4.92  |
| WhiteH w/o PP | 88.74        | 88.05        | 88.05        | 89.20              | 3.30      | 5.02  |
| XD6           | 75.70        | 81.50        | 80.70        | 81.80              | 3.99      | 5.36  |
| Wines         | 85.96        | 85.96        | <b>87.64</b> | 85.96              | 1.21      | 1.75  |
| Zoo           | <b>98.02</b> | 97.03        | <b>98.02</b> | 91.09              | 3.24      | 4.02  |
| Average       | 76.26        | 77.76        | <b>79.85</b> | 78.57              | 3.69      | 5.15  |

thing more interesting appears by looking at the bold-faced results of Table 1. These were chosen to represent excellent results of either the 1-NN, 1-tNN or 1-sNN rules, in the cases where they achieve results at least as good as all other algorithms, including the NN rule ran with the average  $k$ s of the 1-sNN rule. Almost all bold-faced results are in the column 1-sNN, and many of the 1-sNN results are actually bold-faced. If the four algorithms were performing evenly, each would obtain the best result on average  $29/4 \approx 7$  times. A sign test (third column vs all the others) reveals a threshold probability of minute order ( $10^{-27}$ ),



*i.e.* an infinitesimal proportion of configurations appear to be at least as in favor of the 1-sNN as the one presented by Table 1.

All these observations exhibit a real performance of the 1-sNN algorithm. The problem for which the increase is the most dramatic is Lung Cancer. In this case, the use of the 1-sNN rule buys an increase of more than 9% in the accuracy compared to each of the three other rules. Since the 1-sNN builds neighborhoods of larger size compared to 1-NN and 1-tNN, we might conclude that this represent the main reason for the accuracy's increase. Further remarks show that this is a loose explanation. The head-to-head comparison of the 1-tNN and 1-sNN shows that, if we except the atypical Monks 1 dataset, the choice of the 1-tNN instead of the 1-NN buys on average an increasing of 0.4% in accuracy by using 2.69 more neighbors, whereas the choice of the 1-sNN instead of the 1-NN buys on average an increasing of more than 2.5% in accuracy by using only 4.15 more neighbors. Therefore, the difference of less than 1.5 neighbors between 1-tNN and 1-sNN accounts for more than 2% increase in the accuracy, *i.e.* much more than the comparison 1-tNN vs 1-NN. Finally, the  $\bar{k}_{sNN}$ -NN rule, at equivalent neighborhood size, is beaten by  $\approx 1.3\%$  by the 1-sNN rule. This phenomenon can be explained by the fact that the distance between one point voting for another one in the 1-sNN rule is not greater than the largest one in the 1-NN, whereas the largest such distance in the  $\bar{k}_{sNN}$ -NN rule, which uses on average the same number of neighbors as the 1-sNN rule, can be much larger than in the 1-NN, and therefore can provide less reliable votes as distances increase. Altogether, these observations exhibit the more accurate choice of the additional neighbors in the 1-sNN rule compared to the classical NN rules.

## 2.2 Average Comparisons for Many $k$

In this Section, we extend the results of the preceding Section to the cases where the value of  $k$  fluctuates to large values. To compare reliably the three rules  $k$ -NN,  $k$ -tNN and  $k$ -sNN, we choose the following experimental set-up. On each dataset of Table 2, we ran each rule with the values of  $k$  set from 1 to large values  $90 \leq k \leq 100$ . For the datasets with too few examples to reach such large values, the algorithms were ran up to the maximum possible value for  $k$ . The accuracies on each dataset were computed by LOOCV for each possible value for  $k$ .

The first three columns of Table 2 display the respective results of the corresponding NN, tNN and sNN rules. Each result is given as a form (Mean<sub>Standard dev.</sub>, *i.e.*  $\sigma$  is put in index) computed over all runs. These results are useful to compare the relative behaviors of the three rules together, but they are obviously not suited to make performance comparisons with other algorithms: the values of  $k$  ranged towards values far larger than one would fix to maximize the accuracy. A paired sign test was carried out between each of the three accuracy's columns. For the comparison  $k$ -NN vs  $k$ -tNN, it reveals that they cannot be separated up to risk  $\alpha \leq 20\%$ . This, along with the fact that their average accuracies over the 29 datasets are almost identical, shows that the two rules are equivalent. The four datasets shown on Figure 2 bring particularly

**Table 2.** Accuracies on LOOCV for 29 datasets. The average accuracy of sNN is bold-faced because a paired  $t$ -test reveals a threshold risk of order  $< 1/10\%$  when comparing it w.r.t. NN and tNN (see text)

| Dataset       | Accuracy $_{\sigma}$  |                       |                       |
|---------------|-----------------------|-----------------------|-----------------------|
|               | NN                    | tNN                   | sNN                   |
| Audiology     | 65.45 <sub>4.63</sub> | 65.96 <sub>4.48</sub> | 68.79 <sub>2.48</sub> |
| Australian    | 77.89 <sub>0.88</sub> | 77.88 <sub>0.83</sub> | 77.68 <sub>1.27</sub> |
| Balance       | 85.35 <sub>1.11</sub> | 86.48 <sub>1.53</sub> | 86.59 <sub>1.48</sub> |
| Bigpole       | 64.73 <sub>1.23</sub> | 64.75 <sub>1.34</sub> | 65.43 <sub>1.19</sub> |
| Breast-W      | 95.89 <sub>0.38</sub> | 96.03 <sub>0.45</sub> | 96.38 <sub>0.48</sub> |
| Car           | 92.71 <sub>1.46</sub> | 94.86 <sub>1.82</sub> | 94.90 <sub>1.82</sub> |
| Crx           | 78.69 <sub>1.29</sub> | 78.80 <sub>1.13</sub> | 80.67 <sub>0.89</sub> |
| Dermatology   | 54.21 <sub>7.88</sub> | 54.16 <sub>7.31</sub> | 56.51 <sub>6.98</sub> |
| Echo          | 66.51 <sub>1.55</sub> | 67.08 <sub>1.43</sub> | 66.79 <sub>1.83</sub> |
| German        | 72.66 <sub>0.96</sub> | 72.46 <sub>0.08</sub> | 72.57 <sub>1.13</sub> |
| Glass2        | 66.71 <sub>4.95</sub> | 67.91 <sub>2.59</sub> | 69.37 <sub>2.41</sub> |
| Hard          | 49.31 <sub>4.16</sub> | 48.82 <sub>4.78</sub> | 48.97 <sub>5.44</sub> |
| Heart         | 75.32 <sub>1.04</sub> | 75.14 <sub>1.07</sub> | 75.36 <sub>0.89</sub> |
| Hepatitis     | 79.15 <sub>0.64</sub> | 79.26 <sub>5.63</sub> | 79.26 <sub>0.89</sub> |
| Horse Colic   | 68.77 <sub>2.40</sub> | 68.72 <sub>2.25</sub> | 71.16 <sub>2.61</sub> |
| Ionosphere    | 68.50 <sub>6.49</sub> | 68.38 <sub>6.81</sub> | 72.59 <sub>3.21</sub> |
| LED Even      | 89.59 <sub>0.43</sub> | 89.12 <sub>0.27</sub> | 89.18 <sub>0.60</sub> |
| LED Even+17   | 78.58 <sub>1.22</sub> | 79.51 <sub>1.40</sub> | 79.48 <sub>1.38</sub> |
| Lung Cancer   | 25.18 <sub>13.1</sub> | 26.32 <sub>14.4</sub> | 34.07 <sub>16.3</sub> |
| Monks 1       | 79.11 <sub>8.01</sub> | 78.86 <sub>10.0</sub> | 78.86 <sub>8.40</sub> |
| Pima          | 71.24 <sub>0.72</sub> | 70.44 <sub>0.90</sub> | 71.20 <sub>0.64</sub> |
| Tic tac toe   | 72.67 <sub>4.95</sub> | 71.49 <sub>5.78</sub> | 72.77 <sub>8.31</sub> |
| Vehicle       | 72.53 <sub>0.64</sub> | 73.31 <sub>0.56</sub> | 72.85 <sub>0.83</sub> |
| Waves         | 82.10 <sub>1.21</sub> | 82.21 <sub>1.17</sub> | 82.88 <sub>1.57</sub> |
| WhiteHouse    | 89.67 <sub>1.02</sub> | 89.84 <sub>1.10</sub> | 90.86 <sub>1.24</sub> |
| WhiteH w/o PP | 86.94 <sub>0.74</sub> | 86.96 <sub>0.83</sub> | 88.23 <sub>0.79</sub> |
| XD6           | 80.71 <sub>1.17</sub> | 81.33 <sub>1.49</sub> | 80.87 <sub>2.61</sub> |
| Wine          | 55.46 <sub>13.4</sub> | 55.41 <sub>13.6</sub> | 59.02 <sub>13.1</sub> |
| Zoo           | 55.43 <sub>16.5</sub> | 56.54 <sub>17.9</sub> | 60.75 <sub>18.9</sub> |
| Average       | 72.45                 | 72.69                 | <b>73.93</b>          |

nice visual evidence of this fact: the curves of these two rules are quite similar even on the two scales  $k$  and  $\mu(k)$ . Things become different when looking at the  $k$ -sNN rule. A sign test now gives threshold risks of  $\approx 0.0012\%$  and  $0.00005\%$  to separate the  $k$ -sNN rule from respectively the  $k$ -NN and the  $k$ -tNN rule. As shown by Table 2, the primary reason for the test to be significant is of course that the average accuracy is significantly in favor of the  $k$ -sNN rule over the 29 datasets. To explain these results, three reasons can be pointed out. The first two reasons are obviously the same as for the preceding Section. The justification for the better choice of

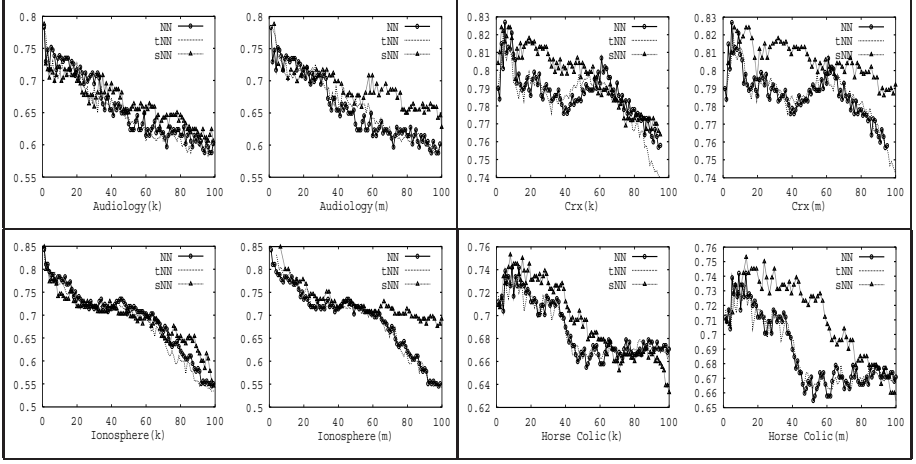


**Fig. 2.** Four representative examples of plots using the two scales on  $k$  and  $\mu(k)$ , respectively noted  $(k)$  and  $(m)$  for each dataset (vertical axes are the accuracies)

the additional neighbors in the  $k$ -sNN algorithm is now visually evident from Figure 2, when looking at the  $\mu(k)$  curves. For the Horse Colic dataset, the curve of the sNN rule is clearly located over the two other curves. On particular points for  $45 \leq \mu(k) \leq 60$ , the accuracies of the NN and tNN rules are similar, but they are beaten by the sNN rule by more than 5% on average, at equivalent neighborhood sizes. The third reason explaining the better results of the  $k$ -sNN rule, which was observed separately for a large majority of the datasets, is its resistance against the accuracy's degradation when the average neighborhood size increases. On Ionosphere (see Figure 2), when  $\mu(k)$  tends to large values, the performances of the sNN rule degrades of about 8% compared to the small values for  $\mu(k) \leq 20$ , whereas for the two other rules, the degradation is greater than 20%. This phenomenon is less pronounced but still visible on the Audiology dataset of Figure 2. It is all the more important for a NN rule to be robust to the variations of  $k$ , as there is no formal rule to fix  $k$ . To conclude this part, we have carried out a paired  $t$ -test on Table 2 to compare each algorithm versus each of the other two over all runs for  $k$ . The limit risk  $\alpha$  for the difference tNN vs NN is only 3.6% (therefore, with a reasonable risk of few percents, we would keep the identity hypothesis between the two algorithms), whereas the two other risks for sNN vs NN and sNN vs tNN are respectively 0.029% and 0.057%; by means of words, we can reject the identity of the performances of sNN and the two algorithms while taking a very small risk  $< 1/10\%$ .

### 2.3 sNN to Fix Pollution Normalization

It is well known that coastal zones are economical areas sensitive to pollution linked to human activities [1]. The administrators of these coastal structures are

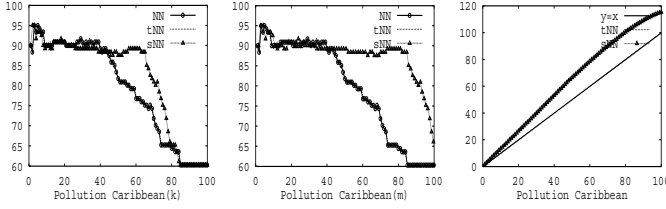


**Fig. 3.** NN, tNN and sNN performances on Caribbean basin pollution data (notations follow Figure 2). Right plot displays curves of  $\mu(k)$  as a function of  $k$

confronted with a crucial problem: What are the toxic metal thresholds required to immerse dredging products in the sea [1]? Experts and research organizations have set up physico-chemical criteria, called grids, which are based on heavy metal concentration levels [1]. These grids used in decision making over long-term storage cannot be applied to various environments since they are basically tailor-made for local applications only. As an example, in the Caribbean basin, comparisons with existing standards (French, English, Canadian, Dutch) establish that all tested areas are safe. Heavy metal (Lead and Zinc) concentrations observed in polluted areas in this region are frequently inferior up to several orders of magnitude with respect to the grids healthy reliable limit. One solution at an affordable price to build previsions with reasonable generalization can be to use ML on local pollution measurements. In this line of work, NN-like rules are a first step in addressing the problem, because of their performances and simplicity. Our data contains 121 instances, each described over 11 attributes (As, Cd, Cr, Cu, Fe, Hg, Mn, Ni, Rb, Sn, Zn), plus a class (Polluted/Non Polluted). The data were collected during the period 1992-1999 in the "Cul-de-Sac Marins" areas (about 300 km<sup>2</sup>) of Guadeloupe which contains in particular a major harbor of the lesser Antilles: that of Pointe-à-Pitre [1]. The results of the LOOCV's comparisons between the three NN rules are displayed in Figure 3. Note that the accuracy of C4.5's tree (default settings) is 91.7%, *i.e.* less than the best values of the NN classifiers. This real-valued dataset is also very interesting to compare our NN rules since the neighborhoods of the NN and tNN rules are the same. The NN and tNN curves are therefore confounded and only the sNN rule can possibly obtain better results. Figure 3 shows that it is indeed the case, as well as does it show the phenomenon observed in Figure 2: the resistance against accuracy's degradation for the sNN rule when  $k$  or  $\mu(k)$  fluctuate.

### 3 Discussion

#### 3.1 About the Efficiency of the sNN Rule

It was recently argued [9] that the curse of dimensionality theoretically rules out nearest neighbors classifiers as reliable classifiers, a thing which is obviously not

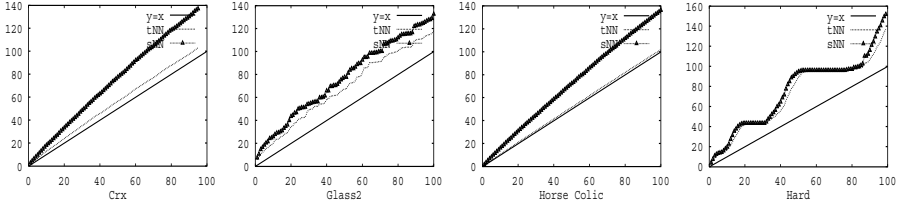


**Fig. 4.** Four examples of plots of the average neighborhood size of the tNN and sNN algorithms as a function of  $k$  (NN's is confounded with  $y = x$ )

true from practice. One possible reason is that instances might be grouped in manifolds whose complexity, and not  $n$ , would guide the curse of dimensionality. The extent that its effects are dimmed for this reason is however unclear [9], but this gives in turn a reason for the success of the 1-sNN rule. Figure 4 plots four examples of  $\mu(k)$  for the tNN and sNN algorithms. Remark that on all examples the curves for sNN are highly distinct from  $y = x$ , whereas the curves for the tNN and  $y = x$  are sometimes very similar, e.g. for the Horse Colic and Crx datasets. Provided  $k$  is not too large, the maximum radius over all neighborhoods, that are the *same* for each rule, remains reasonable enough for the vote of the corresponding instances to be reliable. Since the neighborhoods of the 1-sNN rule contain the greatest number of instances, and because there are more reliable instances to vote, the overall vote will get even better. On the other hand, the Horse Colic dataset (Figure 2) shows that when  $k \geq 90$ , given the greater number ( $> 33\%$ ) of neighbors for the  $k$ -sNN rule getting farer and more unreliable, the  $k$ -sNN rule begins to perform worse than the other rules. Another possible explanation for the good results of the sNN rule could be some smoothness or variation properties over the class conditional densities, stating that on the vicinity of the instances these functions do not fluctuate significantly, or at least they incur the same bias so that predicting Bayes class does not suffer a significant loss (this second hypothesis is also stated in [9]). It is interesting to note that such properties, or something akin to them, were necessary [18] to obtain good theoretical bounds on the error of nearest neighbor rules [4,5,9,17,18]. Obviously, it is not to say that our datasets satisfied such assumptions, even if it might be the case for reasonable hypotheses in real-world domains, such as for our pollution measurements data. However, for example, on simulated datasets for which such reasonable smoothness hypotheses *cannot* be satisfied, the sNN rule did not perform as nicely as usual. This is the case for the XD6 [3], Hard and LED Even [2] domains (see Table 2). Hard was partially generated to theoretically satisfy no such constraint: the target concept is a 5-bits XOR function with 5 additional irrelevant attributes, and attribute noise. In the case of XD6, very weak constraints can be satisfied everywhere in  $\mathcal{X}$  (we refer the reader to [3] for a complete description of the problem).

### 3.2 Speeding-Up the Classification of the sNN Rule

The NN rule has two main complexity-dependent shortcomings. Since it stores all instances in memory, it has a large,  $\mathcal{O}(|\mathcal{S}|)$  storing complexity. Since it must search through all stored instance to classify a new instance, its time complexity is also large,  $\mathcal{O}(|\mathcal{S}|)$ . The sNN algorithm faces the same problems, along with the additional one to find adjuncts in  $\mathcal{S}$ . While instance selection methods [10,11,19] can significantly reduce the storing and classification complexities of the NN and sNN rules, sophisticated data structures were created for the  $k$ -NN rule to reduce the classification complexity of orders, at the expense of a pre-processing stage to organise the reference sample.

The  $kd$ -trees [8,7] are useful tools for fast searching nearest neighbors (in this notation, the “ $k$ ” denotes the dimension of the search space, and *not* the number of neighbors). A multidimensional binary search tree is built, by recursively partitioning the reference sample into subsample of equivalent sizes, according to one current variable, until each leaf contains *buckets* of instances of fixed size  $b$ . This tree is processed by an oriented depth-first search. The search of the  $k$  nearest neighbors of some point  $x$  has two principal components. First, a priority queue maintains the  $k$  nearest instances found so far; second, pruning rules are used to remove parts of the search space to which the true nearest neighbors cannot belong (we refer the reader to [8,7,20] for the detailed algorithm).

As argued by [20], this algorithm is efficient for moderate dimensions. As dimension increases, the repetitive exploration stages in the tree soon visit nearly every bucket, while the tree size consumes more and more space. Two methods of interest for our purposes were proposed to correct this drawback. The first one speeds-up the  $kd$ -trees processing stage [16], by giving to some buckets a class label if their centroid obtains a *confident* classification by the  $k$ -NN rule. Whenever a labeled leaf is reached by an instance to be classified, the class label is immediately given to the instance without any further processing of the tree to search the nearest neighbors. The second one replaces  $kd$ -trees by trees with a different internal node structure [20]: each test is a membership test to an hyperball by defining a *vantage point* (reference point) and specifying a corresponding radius. These trees are called *vp*-trees.

The adaptation of the algorithms to the  $k$ -sNN rule can be made by completing the previous tree structures to find the adjuncts during the pre-processing stage of  $\mathcal{S}$ . After having built these trees for the  $k$ -NN algorithm, for each  $x \in \mathcal{S}$ , we compute the radius of its  $k^{th}$  nearest neighbor,  $r_x$ . We call  $(x, r_x)$  the neighborhood ball of  $x$ . Note that  $\forall y \in \mathcal{S}$ , if  $d(x, y) \leq r_x$ , then  $x$  is one adjunct of  $y$ . After the construction of the chosen tree for the  $k$ -NN rule, we process each of its leaves to make it point to each neighborhood ball intersecting the multidimensional leaf bucket. While it depends on the metric used and on the problem, locally, this number shall remain tractable even while  $\mathcal{S}$  increases, because balls intersecting some bucket will get closer and closer to it. When processing this tree to find the sNN neighbors of some  $x \in \mathcal{X}$ , as soon as we reach the first leaf of the tree, which means that the instance belongs to the corresponding bucket, we test the pointed neighborhood balls to see if the corresponding examples are

adjuncts of  $x$ , and then search for the nearest neighbors of  $x$  by processing the tree as explained in [8,7,20].

### 3.3 Optimizing the $k$ -NN Rule Using the sNN Graph

The principal parameter of the NN rule which can be controlled to optimise the accuracy is  $k$ . Its optimal fixation remains one important theoretical issue in the NN field [15], but two experimental rules have a widespread use. Rule 1 states that we should fix small constant values for  $k$ , such as between 1 and 15. Rule 2 states that we should fix negligible values compared to  $|\mathcal{S}|$ , such as  $k = \sqrt{|\mathcal{S}|}$ . It appears experimentally on a majority of datasets that for sufficiently small  $k$ , the greater the average degree of the sNN graph, the more accurate the  $k$ -sNN rule compared to the  $k$ -NN rule. Table 1 also shows that the fixation of  $k$  to be the average neighborhood size of the 1-sNN rule brings a clear advantage to this  $\bar{k}_{sNN}$ -NN rule compared to the 1-NN rule (in LOOCV, this represents the average neighborhood size of  $\mathcal{S}$  to which the tested instance is removed). We refer to this rule as Rule 3. The comparison Rule 1-3 was carried out on the 29 datasets of Table 1, namely, for each dataset, we have computed the number of values for  $k \in \{1, 2, \dots, 15\}$  for which the accuracy on LOOCV for the corresponding  $k$ -NN rule is greater than that of the  $\bar{k}_{sNN}$ -NN. The average number of values  $k \in \{1, 2, \dots, 15\}$  bringing better accuracies over all problems is 4,67. The median value is 3, which means that the distribution is highly concentrated on the *small* values, or similarly that Rule 3 beats Rule 1 on most cases. To make a simple comparison, replacing the  $\bar{k}_{sNN}$ -NN with the conventional 1-NN leads to much worse results: average 7.5, median 10, *i.e.* a distribution concentrated on the *large* values. The comparison Rule 2-3 shows that the  $\sqrt{|\mathcal{S}|}$ -NN rule performs surprisingly poorly on the 29 datasets, achieving only 76.05% accuracy, a result equivalent to that of the 1-NN rule (see Table 1), but much worse compared to the  $\bar{k}_{sNN}$ -NN rule, whose accuracy is higher by more than 2.5%.

## References

1. D. Bernard. Metals in sediments from two lagoons off guadeloupe, west indies. *Marine Pollution Bulletin*, 30:619–621, 1995. 228, 229
2. C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases. 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>. 224, 230
3. W. Buntine and T. Niblett. A further comparison of splitting rules for Decision-Tree induction. *Machine Learning*, pages 75–85, 1992. 230
4. T. Cover and P. E. Hart. Nearest Neighbor pattern classification. *IEEE Transactions on Information Theory*, pages 21–27, 1967. 230
5. J. A. Drakopoulos. Bounds on the classification Error of the Nearest Neighbor Rule. In *Proc. of the 12<sup>th</sup> International Conference on Machine Learning*, pages 203–208, 1995. 230
6. E. Fix and J. L. Hodges. Discrimatory analysis, nonparametric discrimination. Technical Report TR-21-49-004, Rept 4, USAF School of Aviation Medicine, Randolph Field, TX, 1951. 222

7. J. H. Freidman, J. L. Baskett, and L. J. Shustek. An algorithm for finding nearest neighbors. *IEEE trans. on Computers*, 1975. 231, 232
8. J. H. Freidman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. on Math. Software*, 1977. 231, 232
9. J. H. Friedman. Flexible metric nearest neighbor classification. Technical report, Stanford University Dept of Statistics, 1994. 222, 229, 230
10. G. W. Gates. The Reduced Nearest Neighbor rule. *IEEE Trans. on Inf. Theory*, pages 431–433, 1972. 231
11. P. E. Hart. The Condensed Nearest Neighbor rule. *IEEE Trans. on Inf. Theory*, pages 515–516, 1968. 231
12. D. Harwood, S. Subbarao, H. Hakalahti, and L. S. Davis. A new class of edge-preserving smoothing filters. *Pattern Recognition Letters*, pages 155–162, 1987. 224
13. T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE PAMI*, pages 607–615, 1996. 222, 224
14. R. Meir and R. R. Snapp. Local polynomial models for classification. Draft, 1998. 222, 224
15. S. Okamoto and N. Yugami. Theoretical analysis of the nearest neighbor classifier in noisy domains. In *Proc. of the 13<sup>th</sup> International Conference on Machine Learning*, pages 355–363, 1996. 232
16. A. M. Palau and R. Snapp. The labeled cell classifier: a fast approximation to  $k$  nearest neighbors. In *Proc. of the 14<sup>th</sup> International Conference on Pattern Recognition*, volume 1, 1998. 231
17. R. R. Snapp and S. S. Venkatesh. Asymptotic derivation of the finite-sample risk of the  $k$  nearest neighbor classifier. Technical Report UVM-CS-1998-0101, University of Vermont, Burlington, 1998. 230
18. S. S. Venkatesh, R. R. Snapp, and D. Psaltis. BELLMAN STRIKES AGAIN! the growth rate of sample complexity with dimension for the nearest neighbor classifier. In *Proc. of the 5<sup>th</sup> International Conference on Computational Learning Theory*, pages 93–102, 1992. 230
19. R. D. Wilson and T. R. Martinez. Instance Pruning Techniques. In *Proc. of the 14<sup>th</sup> International Conference on Machine Learning*, pages 404–411, 1997. 224, 231
20. P. N. Yanilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. of the 5<sup>th</sup> ACM-SIAM Symposium on Discrete Algorithms*, 1993. 231, 232



# Automatic Case Base Management in a Multi-modal Reasoning System

Luigi Portinale<sup>1</sup> and Pietro Torasso<sup>2</sup>

<sup>1</sup> Dipartimento di Scienze e Tecnologie Avanzate, DISTA  
Universita' del Piemonte Orientale "A. Avogadro"  
C.so Borsalino 54 - 15100 Alessandria (ITALY)

`portinal@unipmn.it`

<sup>2</sup> Dipartimento di Informatica - Universita' di Torino  
C.so Svizzera 185 - 10149 Torino (ITALY)

`torasso@di.unito.it`

**Abstract.** The definition of suitable case base maintenance policies is widely recognized as a major success key of CBR systems; underestimating this issue may lead to systems that do not perform adequately under performance dimensions, namely computation time, competence and quality of solutions. The goal of the present paper is to analyse an automatic case base management strategy in the context of multi-modal architectures combining CBR and Model-Based Reasoning. The strategy, called Learning by Failure with Forgetting (**LFF**) is based on incremental learning of cases interleaved with off-line processes of case deletion, in order to control the content and the size of the case library. Results from an extensive experimental analysis in an industrial plant diagnosis domain is then reported, showing the usefulness of **LFF** with respect to the maintenance of suitable performance level for the target system.

## 1 Introduction

It is now widely recognized inside the CBR community that the success of a Case-Based system strictly depends on the possibility of devising a suitable case-base management policy; in fact, this means that the problem of knowledge update seems to re-appear inside CBR systems as in any general KBS [7]. Since cases are a form of implicit knowledge, the management of the evolution of this kind of knowledge seems to be, at least in principle, easier to define than for general knowledge. Unfortunately, simple management strategies like adding every case that has been processed to the library do not work in general. Trying to update the library in such a way may cause an overgrowth of the case memory, by significantly impacting the system performance both in terms of computational efficiency and in terms of quality of produced solutions [16,13,14].

On the other hand, more sophisticated solutions like E-MOP organizations of the case memory, while being in principle able to update the knowledge through case generalization and specialization [6], suffer from several problems still resulting in a significant overgrowth of the case library [1,12].

A possible solution to the above mentioned problems is to delegate the management of the case library to the user in a guided way. Indeed, a lot of CBR systems interact suitably with the user after a new case has been solved, in order to allow him/her to decide whether that case has to be stored (and possibly in which form). Particularly interesting in this context are some approaches concerning “case base visualization” techniques that provide the user with an informed view of the case base, where each case may be evaluated by taking into account other cases close to it [17]. Alternatively, the decision of adding a case can be taken by the user by inspecting the results of other classification tools, helping in locating a case in a suitable taxonomy and in evaluating the current competence of the case-base (see for example [3] where a Bayesian classifier is adopted).

In recent years, new approaches based on automatic case base management strategies have been proposed, usually based on a specific competence model of the case base [16,13,18]. In this paper, we propose a rather different approach where the management of the case base is completely transferred to the system, but without the need of any specific competence model. The result is an automatic case base management strategy where user intervention is in principle not required. The context in which such strategy is defined is the ADAPtER system [10], a multi-modal reasoning system combining CBR and Model-Based Reasoning (MBR) for diagnostic problem solving. While the availability of a precise model of the domain and of a model-based solver are crucial to the definition of the strategy, the application task (namely diagnosis) is not so constraining, so we think that this methodology could be applied in other different tasks as well. The case base management strategy we discuss is called **LFF** (Learning by Failure with Forgetting) and has been initially proposed in [14] where its impact on the aspects of computational efficiency, quality of obtained solutions and competence of the case base have been deeply discussed.

In the present paper, we report the results of an extensive experimental analysis we have performed on a real-world case study, showing that the **LFF** methodology is able to control and update the content of the case memory in such a way that: 1) overgrowth will not occur, 2) the competence of the system and the quality of the obtained solutions are maintained at significant levels.

## 2 Architectures for Multi-modal Reasoning

Multi-modal systems relying on CBR are assuming a growing importance in many different application areas. [2,4]. In this paper, we adopt a multi-modal framework where cases are combined with a “deep” model of the domain; in particular, we will consider two slightly different system architectures as follows:

**Architecture A1:** this is the architecture of the system ADAPtER [10,11] using two problem solving components, a CBR one and a MBR one, cooperating through the scheme described by means of the pseudo-code of fig. 1; The problem solving cycle starts by trying to retrieve a case from a case library, easily adaptable to the new case to be solved (RETRIEVE). If no

```

A1(new-case, Case-Memory, Causal-Model, S, T)

IF NOT RETRIEVE(new-case, Case-Memory, S,
                retrieved-solution)
THEN IF MBR(new-case, Causal-Model, mbr-solution, T)
      THEN return(mbr-solution)
      ELSE return("failure")
ELSE
  IF OK-SOLUTION(new-case, retrieved-solution,
                  Causal-Model, replayed-solution)
  THEN return replayed-solution
  ELSE
    IF ADAPTATION(new-case, replayed-solution,
                  Causal-Model, adapted-solution)
    THEN return adapted-solution
    ELSE
      IF MBR(new-case, Causal-Model, mbr-solution, T)
      THEN return(mbr-solution)
      ELSE return("failure")

```

**Fig. 1.** Multi-modal architecture A1 (ADAPtER)

adaptable case can be retrieved, then a model-based solution is computed by the system (MBR). Otherwise, first the system attempts to verify if the retrieved solution can be re-played in the new situation described by the new case (OK-SOLUTION) and, if this step fails, a solution adaptation is tried (ADAPTATION). If no suitable adaptation can be found, then MBR is invoked for solving the problem. Two main parameters are defined in such an architecture: the first is denoted by  $S$  and represents a threshold on the value of a heuristic function evaluating the adaptability effort of the retrieved solution to the current case (see [10,12] for more details); retrieval fails if there is no case in the library for which the heuristics assume a value less than  $S$ . The second parameter is denoted by  $T$  and corresponds to a time limit to the MBR process. If the computational time of MBR exceeds the limit  $T$  without providing a solution, it returns a failure. This is a way of imposing a “practical competence” to MBR, when the theoretical competence is 100% on the problem space (i.e. MBR is in principle able to solve any problem in the domain if the model is complete).

**Architecture A2:** this is a slight revision of A1 suggested by the characteristics of the problems in the considered domains; indeed MBR seems to be able to efficiently solve a large percentage of problems, while it becomes extremely hard to solve some of them (see [9] for some preliminary results). Since the number of “hard” problems is relevant anyway, the use of CBR in the attempt to improve the efficiency of problem solving by exploiting past experience is justified. However, this observation suggests the revision of the

problem solving scheme by trying a “short MBR” first and by resorting to the complete CBR-MBR combination only if the previous step does not provide the solution. Such a scheme is shown in fig. 2. The new parameter  $T'$

```

A2(new-case, Case-Memory, Causal-Model, S, T', T)

IF MBR(new-case, Causal-Model, mbr-solution, T')
  THEN return(mbr-solution)
  ELSE IF NOT RETRIEVE(new-case, Case-Memory, S,
    retrieved-solution)
    THEN IF MBR(new-case, Causal-Model, mbr-solution, T)
      THEN return(mbr-solution)
      ELSE return("failure")
    ELSE
      IF OK-SOLUTION(new-case, retrieved-solution,
        Causal-Model, replayed-solution)
      THEN return(replayed-solution)
      ELSE
        IF ADAPTATION(new-case, replayed-solution,
          Causal-Model, adapted-solution)
        THEN return(adapted-solution)
        ELSE
          IF MBR(new-case, Causal-Model, mbr-solution, T)
          THEN return(mbr-solution)
          ELSE return("failure")

```

**Fig. 2.** Multi-modal architecture A2

( $T' \ll T$ ) represents the time limit accorded to the “short MBR” process.

### 3 Case Base Management

In the problem solving schemata described in section 2 there is no concern about the evolution of the system and in particular to the way the case memory is initialized and updated. The main issue concerns how to provide some form of learning by experience in such a way that the knowledge contained into cases is efficiently preserved and updated in the presence of new (even unexpected) problems.

In our context, the primary mechanism for learning by experience is the ability to learn the solution to a problem through MBR, when the CBR component has been unable to solve it. In such a situation, the addition of the case to the case library fills this particular gap in the competence of the CBR component. However, a learning strategy based just on the addition of cases to the case memory could be inadequate, since it does not take into account the impact the increase of the case memory may have on the performance of the system.

Moreover, there is no guarantee that the learned case will be actually used in the future for solving a new problem via adaptation. In other words, the well-known *utility problem* [15] has to be faced. The strategy we propose, called *Learning by Failure with Forgetting (LFF)* is able not only to learn new cases, but also to selectively forget cases when they are considered no longer useful or dangerous for the performance of the multi-modal system.

### 3.1 Learning Mechanism

We assume that the multi-modal system starts with an empty case memory. In this way, there is no training phase and there is no need for assuming a-priori knowledge about the distribution of problems the system has to solve. Each time a new input case is processed by the system, a time stamp is incremented and the learning module is activated by means of the following rule:

```
IF (NOT RETRIEVE(new-case, Case-Memory, S, retrieved-solution) OR NOT
    ADAPTATION(new-case, replayed-solution, Causal-Model, adapted-solution))
AND MBR(new-case, Causal-Model, mbr-solution, T)
THEN ADD (case-memory, new-case, mbr-solution)
```

This rule captures the failure of the CBR component (either because of the failure of the retrieve step or of the adaptation step) and the success of the MBR component. If the rule is activated, the description of the new case, together with the solution provided by the model-based reasoner is stored into the case memory. If we denote with  $C$  the newly added case, we supplement the system with the following other pieces of information:

$\langle entered, C, t_E \rangle$ ;  $\langle adaptation\_success, C, n_S \rangle$ ;  $\langle adaptation\_failure, C, n_F \rangle$

where  $t_E$  is the time of storage of case  $C$ ;  $n_S$  is the number of adaptation successes for case  $C$  and  $n_F$  the number of adaptation failures of  $C$  (both of them are initialized to 0). These parameters are aimed at taking track of the usefulness of a case after it has been added to the case memory. These kind of statistics are quite simple to collect and the overhead required is negligible.

### 3.2 Forgetting Mechanism

Since the pure addition of cases to the case memory has been proved not quite useful [13,11], it is necessary to devise a forgetting mechanism for limiting the growth of the case memory and, more importantly, for maintaining in the case memory cases which actually help in solving new problems and allow to save computational effort.

Informally a case is useless when it has either never been retrieved or never been adapted with success, despite the fact it has been in memory for a significant period of time. In particular, the fact that a case has been retrieved and caused a subsequent failure in adaptation is more important and critical than the situation in which a case has never been retrieved. The former situation represents a potentially dangerous situation: the presence of *false positives* in the library (i.e. the presence of cases that may appear useful for solving a problem but that are

actually not) . This may cause a significant slow-down during problem solving and potential problems also for competence. While for never retrieved cases the memory permanence time is obviously an important parameter (less time the case has spent in memory, less probable is in general its retrieval), for cases generating adaptation failure it should be important to detect them as soon as possible, without considering how much time they have spent in memory.

Since forgetting may potentially involve the analysis of every case stored in the library, it is clear that such an activity should be done only occasionally, in order to maintain low overhead. First of all, a time window of variable width  $\tau$  is introduced for evaluating the usefulness of a case and for determining at what time point the forgetting mechanism is activated.

**Definition 1.** *A stored case  $C$  is useless at forgetting time  $t$  if there exist items  $\langle \text{entered}, C, t_E \rangle$  with  $t_E < t - \frac{\tau}{2}$  and  $\langle \text{adaptation\_success}, C, 0 \rangle$ .*

**Definition 2.** *A stored case  $C$  is a false positive if there exist items  $\langle \text{adaptation\_failure}, C, n_F \rangle$  with  $n_F > 0$  and  $\langle \text{adaptation\_success}, C, n_S \rangle$  with  $n_F > n_S$ .*

If a forgetting is triggered at time  $t$ , the following pseudo-code is executed.

```

FOR EACH case C in the case memory
  IF useless(C,t) or false-positive(C,t)
    THEN remove(C,Case-memory)
  ELSE
    set <adaptation-success, C, 0>
    set <adaptation-failure, C, 0>
    set <entered, C, t>

```

The forgetting mechanisms invoked at time  $t$ , removes all the useless and false positive cases. It also updates the statistics by resetting the relevant parameters. This implies that the evaluation of the usefulness of the cases may restart for another time window, by considering all the cases as being learnt at time  $t$ . The other critical parameter is the width  $\tau$  of the time window. It appears reasonable to link the occurrence of the forgetting process to the size of the case memory. Moreover, it seems also reasonable to restructure the memory only when the system exhibits a stable behavior. This means that a minimum number of cases has to be examined from the last forgetting and that the *failure rate*, defined as the percentage of cases that have not been solved by the CBR component in a given time window, must not be too high. **LFF** obtains this goal by triggering forgetting when the current time stamp  $TS$  satisfies the following condition:

$$TS > \max(\delta, k|CM|)$$

where  $\delta$  and  $k$  are positive constants and  $|CM|$  is the size (cardinality) of the case memory. The parameter  $\delta$  is used to assure that, even if the case memory has a rapid growth (as is the case in the initial phase of the life of the system, in particular if an empty initial memory is assumed), forgetting does not take

place for a suitable time interval (i.e. at least  $\delta$  cases have to be examined from the last forgetting). Every time a forgetting occurs, the time-stamp  $TS$  is reset.

Since **LFF** learns a new case when the MBR component is invoked, the width  $\tau$  of the current time window resulting from the above criterion is related to the *failure rate* exhibited in such a window by the CBR component. Given a period of observation of  $N$  cases, if the CBR component exhibits  $M$  failures in such a period, we define the *failure rate* as  $f_r = \frac{M}{N}$ .

Parameters  $f_r$  and  $\tau$  can be formally related, since it can be proved that

$$N > k|CM_N| \text{ if and only if } f_r < \frac{1}{k} - \alpha$$

where  $|CM_N|$  is the size of the case memory at time  $N$  and  $\alpha$  is a positive constant depending on the size of the initial case memory (see [14]).

This means that a forgetting will occur in a given time window if and only if the failure rate is less than a quantity that is less of the inverse of  $k$ . It formally proves that the constant  $k$  is inversely related to the failure rate; this means that if a low failure rate is expected, we can set a large value for  $k$ , by letting the system working without forgetting for a significant amount of time and with a limited growth of the case memory. Of course, if we set a value of  $k$  that is too large for the exhibited failure rate, an overgrowth of the case memory will occur, since the satisfaction of the condition for triggering the forgetting will tend to be postponed. On the contrary, setting a value of  $k$  too small with respect to the actual failure rate will have the effect of an excessive occurrence of forgetting, by producing the equivalent of *thrashing* in a paged operating system.

## 4 System Performance and Experimental Results

### 4.1 Strategy Evaluation

The evaluation of the **LFF** case management strategy has to be defined with respect to particular parameters. As reported in [14], the main motivation for having proposed **LFF** was the analysis of the utility problem in the context of a multi-modal architecture using CBR. Performance dimensions may be identified by considering: *computational efficiency*, *quality* of produced solutions and *competence* of the problem solver. In a CBR context, a good case base management can improve some aspects of performance by tuning the content of the case base. However, the utility problem makes hard (or even impossible) to obtain general improvements over all the above dimensions. In the following we will show that **LFF** can provide satisfactory results over all performance dimensions, by being able to control the utility problem. We will do this by reporting results of an extensive experimental analysis on the two multi-modal architecture A1 and A2 defined in section 2.

In defining the experimental setting, we considered:

- the domain(s) used as test-bed;
- the test sets used for the experimental analysis;

- the formal definition of the performance dimensions;
- the setting of the controlled parameters which can influence the performance of the system.

Concerning the domain, we concentrated on experimental results obtained on a real-world domain concerning the diagnosis of faults in an industrial plant. Such a domain allowed us to generalize some previous results obtained on smaller and simpler domains concerning car engine fault diagnosis and the diagnosis of the leprosis disease (see [11]).

In order to evaluate the performance of the system in dependence of various architectural choices, we need to have precise knowledge and control on the cases of the test set. For this reason we have created different test sets by running a simulator we have developed. The characteristics of a batch of cases automatically generated by the simulator can be controlled by means of suitable parameters; the most significant one we have used is the maximum number of injected faults.

In particular, we considered three different test sets:

- $TS_1$  containing 2500 cases automatically generated by injecting up to 3 faults
- $TS_2$  containing 1000 cases automatically generated by injecting up to 2 faults
- $TS_3$  containing 1500 cases automatically generated by injecting up to 4 faults.

In some experiments we have considered also the test set  $TS_4$  containing 1268 cases. The cases included in  $TS_4$  are the ones of  $TS_1$  which have at most two alternative diagnoses (that is, the case description of these cases includes a number of observations which are sufficient for an almost perfect discrimination).

Concerning the definition of performance dimensions, it is quite natural to estimate *computational efficiency* by means of the average computation time for solving a given problem and to estimate *competence* of the problem solver in terms of the percentage of cases that can be solved in a given batch. In this paper we do not directly consider the analysis of the computation time (see [14] for some results on this parameter), but rather we concentrate on the case memory size parameter; this is a more direct measure of the suitability of the case base maintenance methodology **LFF** and is directly related to computation time (i.e. larger is the set of cases, higher tends to be the retrieval time and consequently the overall computation time).

Defining *solution quality* is more subtle. In such a case the application task becomes quite relevant; since our task is diagnostic problem solving, we decided to estimate this parameter by resorting to a classical notion of quality in diagnosis, i.e. *minimality* (in terms of assumed faults) of the solution. We adopted a very restrictive measure of quality, namely the percentage of cases for which the multi-modal system provides exactly the set of minimal solutions. We recognize the restrictiveness of this definition, but this allows us to have a rather pessimistic analysis, subject to improvements by relaxing the quality measure.

The main thresholds and parameters influencing the behavior of the diagnostic system are the time limits  $T$  and  $T'$ , the adaptation thresholds  $S$  and the learning parameters  $k$  and  $\delta$ . Concerning time limits, we have set thresholds  $T = 60\text{sec}$  and  $T' = 150\text{msec}$  on the CPU time of a Sun Sparc 20, while we



adopted an adaptation threshold  $S = 30$  that may be considered a medium one in the current domain. The **LFF** strategy has then been tested, on the different test sets, by using  $k = 3$  and  $\delta = 300$  (i.e. the case memory size is scaled up 3 times for triggering the forgetting and at least 300 cases are examined from the last forgetting).

## 4.2 Results

The experiments considered the automatically generated test sets  $TS_1$ ,  $TS_2$  and  $TS_3$  for a total of 5000 cases, with different characteristics in terms of problems described, captured by the different number of injected faults in the different test sets as mentioned before. Fig. 3 shows how the **LFF** strategy controls the case library size while processing the cases in the batch. We obtained a case

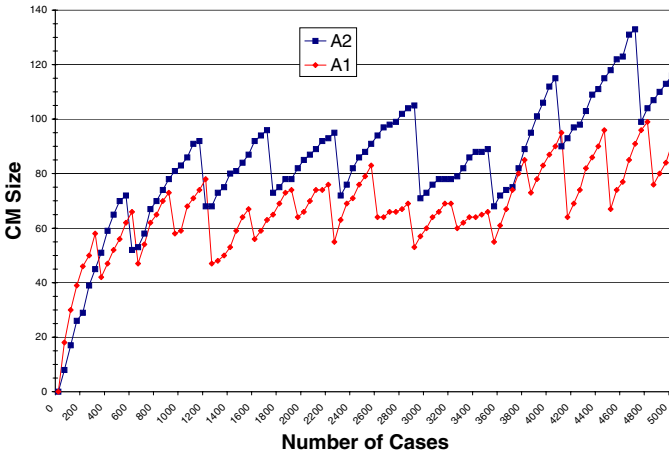


**Fig. 3.** Case Memory Size vs. Number of Examined Cases

memory with about 120 cases on architecture A2 and about 90 cases on A1, after the system has examined all the 5000 cases. The **LFF** strategy appears then to be able to actually take under control the growth of the case memory, even in presence of significant changes in the characteristics of examined cases during problem solving ( $TS_2$  reduces the injected faults wrt to  $TS_1$ , while  $TS_3$  augments them). Following the terminology introduced in [8], **LFF** appears to control in a suitable way significant changes in *problem distribution regularity*.

The second part of experiments concerns the evaluation of **LFF** wrt to competence and solution quality. Results are summarized on fig. 4 and fig. 5 respectively. Results shows a satisfactory behavior of **LFF** both with respect

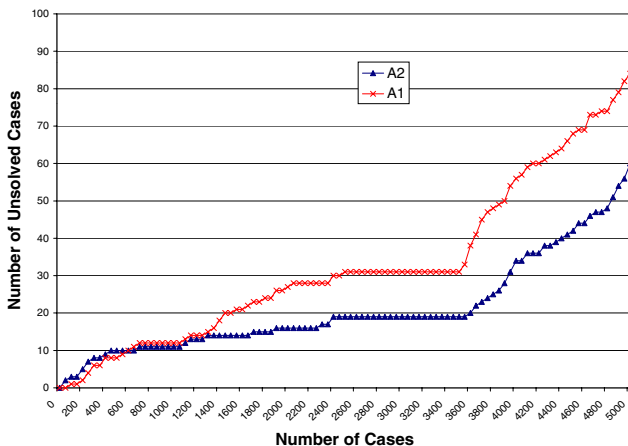


**Fig. 4.** Competence vs. Number of Examined Cases

to competence and to quality, by also putting in evidence a better performance on architecture A2.

It is worth remember that quality results are actually obtained by using a very restrictive measure of quality; however, it is worth noting that if we analyse only cases for which up to 2 solutions are available (i.e. test set  $TS_4$  obtained from  $TS_1$  as mentioned before) we obtain very satisfactory results also by using this measure (96% for A2 and about 90% for A1).

## 5 Conclusions

In the present paper, we have presented the experimental analysis of a case base management strategy in the context of multi-modal reasoning systems combining CBR with MBR. The strategy, called **LFF** is quite effective, since it is able to maintain in the case memory a relatively small number of cases that are actually quite useful, in order to maintain a suitable performance level along considered performance dimensions. Following terminology introduced in [7], this strategy belongs to the category of *diachronic strategies* with respect to

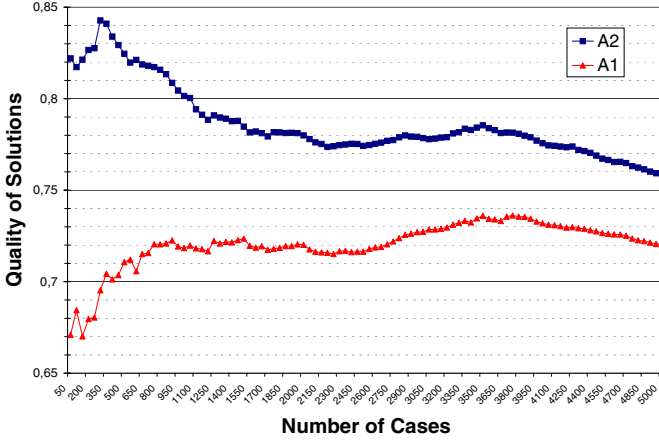


**Fig. 5.** Solution Quality vs. Number of Examined Cases

*data collection* dimension, since it tries to identify trends in case usage over time. With respect to *execution* dimension, it can also be categorized as having *conditional activation timing* (the forgetting condition for case removal and CBR failure for case addition), *off-line integration type* (it works off-line with respect to the problem solving cycle) and with a *broad scope of changes* (it involves the whole case library). Of course, the multi-modal framework slightly changes the context with respect to pure CBR systems, however the main issues, concerning the control of the utility problem and the knowledge management and evolution are essentially the same. The proposed strategy takes a different approach with respect to “competence-based” techniques like the ones described in [16,18], where a specific competence model is maintained in terms of solvability of a target case with respect to the cases in the case memory. However, preliminary results show that in our context, even if a competence-based strategy can be devised (see [13]), no clear advantages seem to emerge. This is particularly true if one consider that the implementation of **LFF** is very simple and does not require any explicit competence model to be updated. It is worth noting that the methodology we have reported in the paper is not restricted to the diagnostic task. It could be quite useful also for planning, since diagnosis and planning share a lot of commonalities. In both tasks, domain theory plays an important role and it can be used for solving problems from scratch. However, both in planning and in diagnosis, first-principle problem solvers may be unable to solve some problems, because of computational complexity and the adaptation of already solved problems can be an effective alternative [5,14]. In our work, we have

shown that the advantages of this integration can be achieved without requiring the user to initialize the case memory or to maintain it. Finally, it is worth remember that **LFF** is effective in controlling changes in *problem distribution regularity*, while we did not address the *problem solution regularity* aspects [8]; interventions on the heuristics used for our adaptation-guided retrieval should be studied in order to address this issue.

## References

1. D. S. Aghassi. Evaluating case-based reasoning for heart failure diagnosis. Technical report, Dept. of EECS, MIT, Cambridge, MA, 1990. 234
2. D. Aha and J. Daniels (eds.). *Proc. AAAI Workshop on CBR Integrations*. AAAI Press, 1998. 235
3. R. Bellazzi, S. Montani, and L. Portinale. Retrieval in a prototype-based case library: a case study in diabetes therapy revision. In *Proc. 4th EWCBR, LNAI 1488*, pages 64–75. Springer Verlag, 1998. 235
4. E. Freuder (ed.). *AAAI Spring Symposium on Multi-modal Reasoning*. AAAI Press, 1998. 235
5. A. Gerevini and I. Serina. Fast planning trough greedy action graphs. In *Proc. AAAI 99*, pages 503–510, 1999. 244
6. J. L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993. 234
7. D. B. Leake and D. C. Wilson. Categorizing case-base maintenance dimensions and directions. In *Proc. 4th EWCBR, LNAI 1488*, pages 196–207. Springer Verlag, 1998. 234, 243
8. D. B. Leake and D. C. Wilson. When experience is wrong: examining CBR for changing tasks and environments. In *Proc. 3rd ICCBR, LNAI 1650*, pages 218–232. Springer Verlag, 1999. 242, 245
9. D. Magro and P. Torasso. Performance issues in unimodal and multimodal reasoning approaches to diagnosis. Technical report, Dip. di Informatica, Univ. di Torino, 2000. 236
10. L. Portinale and P. Torasso. ADAPtER: an integrated diagnostic system combining case-based and abductive reasoning. In *Proc. 1st ICCBR, LNAI 1010*, pages 277–288. Springer Verlag, 1995. 235, 236
11. L. Portinale and P. Torasso. Performance issues in ADAPtER a combined CBR-MBR diagnostic architecture. In *Proc. AAAI Spring Sympos. on Multi-Modal Reasoning*. AAAI Press, Stanford, 1998. 235, 238, 241
12. L. Portinale, P. Torasso, and D. Magro. Selecting most adaptable diagnostic solutions through Pivoting-Based Retrieval. In *Proc. 2nd ICCBR, LNAI 1266*, pages 393–402. Springer Verlag, 1997. 234, 236
13. L. Portinale, P. Torasso, and P. Tavano. Dynamic case memory management. In *Proc. ECAI 98*, pages 73–78, Brighton, 1998. 234, 235, 238, 244
14. L. Portinale, P. Torasso, and P. Tavano. Speed-up, quality and competence in multi-modal case-based reasoning. In *Proc. 3rd ICCBR, LNAI 1650*, pages 303–317. Springer Verlag, 1999. 234, 235, 240, 241, 244
15. B. Smyth and P. Cunningham. The utility problem analysed: a case-based reasoning perspective. In *LNAI 1168*, pages 392–399. Springer Verlag, 1996. 238
16. B. Smyth and M. T. Keane. Remembering to forget. In *Proc. 14th IJCAI*, pages 377–382, Montreal, 1995. 234, 235, 244

17. B. Smyth and E. McKenna. Modeling the competence of case-bases. In *Proc. 4th EWCBR, LNAI 1488*, pages 208–220. Springer Verlag, 1998. [235](#)
18. B. Smyth and E. McKenna. Building compact competent case-bases. In *Proc. 3th ICCBR, LNAI 1650*, pages 329–342. Springer Verlag, 1999. [235](#), [244](#)

# On Quality Measures for Case Base Maintenance

Thomas Reinartz<sup>1</sup>, Ioannis Iglezakis<sup>1</sup>, and Thomas Roth-Berghofer<sup>2</sup>

<sup>1</sup> DaimlerChrysler AG, Research & Technology, FT3/AD,  
P.O. Box 2360, 89013 Ulm, Germany

{ioannis.iglezakis,thomas.reinartz}@daimlerchrysler.com

<sup>2</sup> tec:inno GmbH,

Sauerwiesen 2, 67661 Kaiserslautern, Germany

thomas.roth-berghofer@tecinnno.com

**Abstract.** Case base maintenance is one of the most important issues for current research in Case-Based Reasoning (CBR). In this paper, we outline two novel steps as part of the maintenance phase of the CBR process. The review step covers assessment and monitoring of the knowledge containers whereas the restore step actually modifies the contents of the containers according to recommendations resulting from the review step. Here, we focus our attention on the review step for the case base. For this purpose, we define several quality measures based on different case and case base properties that describe specific characteristics of the case base such as correctness, consistency, uniqueness, minimality, and incoherence. These measures allow an initial implementation of the review step for the case base container. We conclude the paper with an outline of future work to extend these aspects of maintenance in CBR.

## 1 Introduction

During the last decade, Case-Based Reasoning (CBR) evolved from initial ideas originating in cognitive science to a well established intelligent technology suitable to support various applications. One of the consequences is that the focus of current CBR research has moved from issues of case base modeling and acquisition, retrieval and indexing tasks, and similar basic challenges more and more towards application and practice oriented goals. In particular, the early research focus on creating an initial case base is now of less interest but what to do with the contents of a case base over time has become one of the crucial questions in CBR today.

First papers on these issues called efforts along these lines Case Base Maintenance (CBM) (e.g., see [4]). From our point of view, the overall concept of CBM includes all tasks that occur during the lifetime of a CBR system. However, a general framework that describes all the tasks in CBM in a unique way is not yet available.

In this paper, we focus our attention on CBM activities that affect the case base rather than the vocabulary, similarity, or adaptation knowledge containers [8]. For simplicity, we consider any case base as a set of cases — currently,

we do not think about indexing schemes, specific representational issues, or similar additional aspects related to the case base.

In practice, a case base as a set of cases changes over time. New cases are added, old or invalid cases are deleted, similar cases are combined to more general cases, conflicting cases are corrected and so on. All of these changes only happen if some kind of indicator invokes the respective mechanisms to change the case base. Therefore, we have to define means of quality control that enable specific realizations of such indicators. In this paper, we argue that quality measures based on basic properties of single cases and sets of cases are able to implement strategies for quality control in CBR.

This paper is organized as follows. In the next section, we report on some preliminary considerations related to quality control in CBR and briefly describe two novel steps as part of the maintenance phase of the CBR process — these steps are the review and the restore step, and here we focus our attention on the review step. Then, we define the notations necessary to specify quality measures. Thereafter, we describe and define various properties of single cases and sets of cases. These properties are then utilized to define quality measures in section 5. We close the paper with concluding remarks and issues for future work.

## 2 Quality Control for Case-Based Reasoning

Since early definitions of the CBR process which emphasized the application cycle of CBR, it is now more and more accepted that there is an additional maintenance cycle in CBR needed to cope with all issues that arise when CBR is used in practical applications and especially when the case base contents change over time [3]. We view the first three REs in the current state-of-the-art standard CBR process [1] — Retrieve, Reuse, Revise — as the application cycle whereas the fourth RE — Retain — is the first step in the maintenance phase of CBR.

Initially, the difference between application and maintenance originates from the separation of tasks for using the case base (i.e., find appropriate existing experiences and reuse them for novel situations) and additional tasks to keep the case base in a usable state. However, current research in the area of CBM does not sufficiently define what this usable state looks like and which criteria help to detect situations with less usable case bases.

In software engineering, such usability aspects of a program or system during the stages of its development as well as later when the program or system is used in practice are considered by quality control. This is the initial motivation to consider quality control in CBR, too. Obviously, there is an additional aspect beyond quality control that enables actions to improve the quality of a program or system when the control mechanisms indicate that the quality is not appropriate.

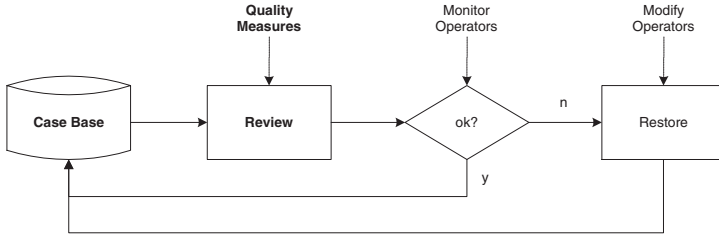


**Fig. 1.** Maintenance steps in CBR

## 2.1 Maintenance Steps in CBR

These comparisons and ideas lead to the definition of two novel steps as part of the maintenance phase of the CBR process. Figure 1 shows the review and the restore steps in CBM along with the control flow between them. The two steps start with a case base as their input. This case base is either the result of an initial knowledge acquisition step at the beginning of a CBR project or the outcome of modifications to the case base at a certain stage after the case base is already in use.

The review step considers the current state of the case base and assesses its quality. For this purpose, we have to define quality measures that allow the computation of values that indicate the current quality of the case base. These quality values are then monitored and specific indicators lead to the initiation of the restore step.

The restore step utilizes specific modify operators to change or to adapt the contents of the case base. In an ideal setting, the review step already suggests specific changes to return to the level of quality desired. If there is no need to go to the restore step since the quality values are still in good shape, we simply return to the case base, keep it unchanged, and start with the next iteration of the maintenance cycle.

All in all, we propose a new 6RE CBR process which consists of six steps: Retrieve, Reuse, Revise, Retain, Review, and Restore. The first three steps form the application (or usage) cycle whereas the last three build the maintenance phase in CBR.

The two maintenance steps in figure 1 completely abstract from other issues in maintenance of CBR systems. We are aware of the fact that additional maintenance tasks are important if we work on the maintenance task in a more complete way. For example, we also have to consider the retrieval behavior and the appropriateness of the similarity measures over time. However, we argue that it is important to start with the case base as the central knowledge container in CBR, and that it is also important to start with reasonable chunks of tractable subtasks in maintenance.



Moreover, we focus our attention on the review step in this paper and consider the restore step as a topic for future work and further considerations.

## 2.2 Related Work

If we consider existing work towards quality measures for CBR that also consider the case base, we are aware of the following references.

For example, Aha and Breslow [2] use measures suggested by CBR vendors to optimize conversational CBR systems.

In the area of unstructured case bases, Racine and Yang [7] give a definition for inconsistency and redundancy. Inconsistency can be detected through background knowledge or sections which contradict each other and redundancy is observed through subsumption. These measures are limited to the underlying CBR approach.

More general measures are defined by Leake and Wilson [5]. They use measures (regularities) to define two kinds of relationships. First, they define relations between the similarity of problems and the similarity of solutions. Second, they define relations between prior problems and new problems.

Racine and Yang [6] suggest different measures (criteria) to evaluate case bases. However, their focus of interest is consistency — they define measures for intra-case inconsistencies and inter-case inconsistencies for consistency management and the optimization of validation methods. Intra-case inconsistency is defined as a case property which deals with constraint violations of attributes within a case. Inter-case inconsistency is defined as a case base property which can be found across two or more cases.

Smyth and Keane [9] optimize the contents of the case base to preserve the competence of the case base through case deletion. They define the two measures coverage and reachability to achieve this. Coverage of a case is the set of all problems in the problem area which can be solved by this case through adaptation. Reachability of a case is the set of all cases which are used to solve this case through adaptation. Coverage and reachability cannot be calculated because the possible set of problems is in general too vast. Thus the assumption is made that the problem distribution in the case base is representative and a heuristic is used.

Further work of Smyth and McKenna [10] brings up the measures of case density and group density for modeling competent CBR systems. Another strategy of competence preserving is given by the definition of the measure of coverage from Zhu and Yang [11]. Both strategies are case addition rather than case deletion policies.

In summary, all of these existing measures are limited to the underlying CBR approaches, and they use or affect other knowledge containers like the similarity container or need heuristics to be calculated.

### 2.3 Requirements for Quality Measures in CBR

In conclusion, the initial thoughts on quality control in CBR and the review of the relatively few existing papers on quality measures for CBR lead to the following requirements on quality measures.

First, the quality measures and their values have to reflect user requirements that usually ask for a small, comprehensive, and consistent case base. Second, these measures should be simple in the sense that their computation is possible in practice. Third, we also expect the measures to be application independent and as system independent as possible. Although specifics of quality control are possibly context dependent, the general concepts of quality control should be valid across all applications and systems.

## 3 Notation

In this section, we define notations to represent cases and sets of cases. These definitions are necessary to later specify concrete quality measures for CBR. For all notations, we aim at as general definitions as possible with maximum flexibility to cover cases and case bases in most CBR applications and systems.

We start the notations with the basic components for building cases, namely attributes, values, problems, and solutions (see definitions 1 to 3). We choose an attribute–value representation since it ensures high flexibility and generality. We are able to transform most case representations into an attribute–value representation if necessary.

**Definition 1 (Attributes and Values).** *An attribute  $a_j$  is a name accompanied by a set  $V_j := \{v_{j1}, \dots, v_{jk}, \dots, v_{jN_j}\}$  of values. We denote the set of attributes as  $A := \{a_1, \dots, a_j, \dots, a_N\}$ , and the set of values as  $V := \bigcup_{j=1}^N V_j$ .*

Each attribute consists of an identifier (the name) and a set of possible values. For simplicity, we assume for all attributes, especially for quantitative attributes, that the set of values contains only "occurring" values. This means by definition and the dynamic nature of CBR that sets of values are dynamic as well. For example, adding a new alternative answer to a question in conversational CBR means extending the set of values for this specific attribute.

Now, we define problems as sets of attribute values and solutions as any form of information that contributes to the solution of a given problem. Note, we generally assume that the single fault assumption holds, i.e., a single problem description only covers a single problem which in turn requires a single solution. If we encounter a case which covers more than a single problem, we have to transform this case into more than one case until the single fault assumption holds again. However, we do not further elaborate this issue here.

**Definition 2 (Problem).** *A problem is a set  $p_i := \{p_{i1}, \dots, p_{ij'}, \dots, p_{iN_i}\}$  with  $\forall j' \in [1; N_i] \exists a_j \in A \exists v_{jk} \in V_j : p_{ij'} = v_{jk}$ , and  $\forall j \in [1; N] : |(p_i \cap V_j)| \leq 1$ . We denote the set of problems as  $P := \{p_1, \dots, p_i, \dots, p_M\}$ .*

The first condition in definition 2 ensures that for each element in the set of values of a problem there exists a corresponding attribute and a respective value in its set of values. The second condition makes sure that for each attribute a problem does not contain more than a single value. The latter assumption simplifies from situations where it makes sense to allow more than a single value for specific attributes. However, this situation is either emulated by adding extra sets of single values to the set of values or by relaxing the second condition at a later stage of research.

This particular set-oriented definition enables easy implementation of the quality measures defined below. Note, we do not assume that a single problem specifies values for each available attribute. In contrast, a problem only contains those values for attributes which are relevant to describe the specific situation at hand.

**Definition 3 (Solution).** *A solution is any item  $s_i$ . We denote the (multi-) set of solutions as  $S := \{s_1, \dots, s_i, \dots, s_M\}$ .*

For a solution, we do not define any additional requirements in order to avoid restrictions for solutions in any way (see definition 3). For example, a solution possibly contains any text or media information which contains a description how to "solve" the problem. The more structured the domain is, the more structured information or definition we provide for a solution. For example, in classification domains, a solution is simply yet another attribute with a domain of values containing all occurring class labels. On the restrictive side of this flexibility, we are only able to think about solutions in terms of match or mis-match but not in terms of any other more fine-grained meanings.

For both sets, the set of problems and the set of solutions, we assume that they contain  $M$  elements. Since we are aware of the fact that the same solution possibly solves different problems, we allow the set  $S$  of solutions to contain the same element more than once. Hence,  $S$  in definition 3 is possibly a multi-set. We further assume that the enumeration of problems and solutions in  $P$  and  $S$  corresponds to each other, i.e., the solution for a given problem has the same index in  $S$  as the problem has in  $P$ .

For the moment, we also presume that  $P$  and  $S$  contain components of "real" cases, i.e., cases that (currently) exist in the case base and that make sense in terms of the application domain. If the contents of the case base change, both sets  $P$  and  $S$  change, too. In addition to such real cases, we also have potential cases (i.e., cases which can occur by combining any possible combination of attribute values) and meaningful cases (i.e., potential cases that in fact lead to a case that corresponds to any possible case in the application domain that makes sense). For example, occurring queries to the CBR system provide initial hints for such meaningful cases.

**Definition 4 (Case and Case Base).** *A case is a tuple  $c_i := (p_i, s_i, q_i)$  with a problem  $p_i$ , a solution  $s_i$ , and additional quality information  $q_i$ . A case base is a set of cases  $C := \{c_1, \dots, c_i, \dots, c_M\}$ .*

**Table 1.** Six cases from an example case base

|       | $p_i$    |          |          |          | $s_i$ | $q_i$ |
|-------|----------|----------|----------|----------|-------|-------|
| $c_1$ | $v_{12}$ | $v_{23}$ | $v_{31}$ |          | $s_1$ | $q_1$ |
| $c_2$ | $v_{13}$ | $v_{23}$ |          |          | $s_2$ | $q_2$ |
| $c_3$ | $v_{13}$ | $v_{23}$ |          |          | $s_3$ | $q_3$ |
| $c_4$ | $v_{13}$ | $v_{23}$ | $v_{32}$ | $v_{41}$ | $s_4$ | $q_4$ |
| $c_5$ | $v_{13}$ | $v_{23}$ | $v_{32}$ |          | $s_5$ | $q_5$ |
| $c_6$ | $v_{12}$ |          | $v_{31}$ | $v_{45}$ | $s_6$ | $q_6$ |

Definition 4 states that a case is a tuple containing a problem component  $p_i$ , a solution component  $s_i$ , and an additional information component  $q_i$ . This additional information component is comprised of any extra information that is necessary for quality control and that is related to the life cycle of this specific case. For example, time stamps for the time the case is added to the case base and the time of its last access or the relative number of correct usages of the case are typical information items. In this paper, we do not consider this additional information but focus on the problem and solution components of cases.

In comparison to other definitions and representations of cases, we do not distinguish between problem — as an initial short characterization of the problem — and situation which is then a more detailed description of the context of the entire challenge. We omit this distinction to keep the representation as simple as possible. If we utilize this kind of representation in an application domain where there typically is an initial short problem characterization and then a more detailed situation description, we simply assume that the final problem component in definition 4 already contains both, the problem and the situation, possibly acquired during a dialogue or by asking for more detailed symptoms through tests.

*Example 1 (Set of Cases).* Assume a domain with  $A := \{a_1, a_2, a_3, a_4\}$ ,  $V_1 := \{v_{11}, v_{12}, v_{13}\}$ ,  $V_2 := \{v_{21}, v_{22}, v_{23}, v_{24}\}$ ,  $V_3 := \{v_{31}, v_{32}\}$ , and  $V_4 := \{v_{41}, v_{42}, v_{43}, v_{44}, v_{45}\}$ .  $G := \{c_1, c_2, c_3, c_4, c_5, c_6\}$  in table 1 is a set of cases in this domain.

## 4 Case and Case Base Properties

In this section, we present several case and case base properties. Case properties describe characteristics of single cases whereas case base properties provide information about sets of cases, i.e., subsets of cases in the case base. We define case properties as the atomic concepts to specify case base properties which in turn form the basis for the definition of quality measures for CBR. In particular, atomic properties are independent of each other, i.e., there is no general relation between any pair of two properties.

For case properties, we further distinguish between isolated and comparative characteristics. Whereas isolated properties only consider characteristics of a

single case to define properties of that single case, comparative properties also take into account information on other cases and compare them to the single case considered. Note, comparative case properties still define only characteristics of a single case although they consider more than a single case.

#### 4.1 Isolated Case Properties

The most important isolated case property is correctness. We consider a case as correct if the given solution component really "solves" the problem specified by the problem component (see definition 5). The crucial part of this definition is the notion "solves". At this point, we basically use a place holder for any type of relation "solves" denoted by  $\mathcal{S}$ . Thereby, the definition is open to all application domains. For example, in traditional classification domains, the solution of a case is a class label and the relation  $\mathcal{S}$  between problem and solution holds if this class label corresponds to the true class label of the problem. Note, as soon as we want to compute the correctness of a case, we have to define relation  $\mathcal{S}$  precisely.

**Definition 5 (Correctness).** *Assume a binary relation  $\mathcal{S} \subseteq P \times S$ , and a case  $c_i \in C$ .  $c_i$  correct  $:\Longleftrightarrow \mathcal{S}(p_i, s_i)$ .*

For the moment, we do not define any other case properties that only consider characteristics of a single case although we expect that the additional information component  $q_i$  of a case naturally leads to more isolated case properties. For example, the information component  $q_i$  enables control of time and usage aspects and is therefore able to indicate the necessity of maintenance based on these aspects. In this paper, we consider the behavior of a single case in relation to other cases in the case base as more important, and we will now focus our attention on this situation.

#### 4.2 Comparative Case Properties

In this subsection, we turn to comparative case properties that describe characteristics of single cases in relation to other cases. For all of the following definitions in this subsection, we assume that each case is correct.

**Definition 6 (Consistency).** *Assume  $G \subseteq C$ , and  $c_i \in G$ .  $c_i$  consistent within  $G$   $:\Longleftrightarrow \nexists c_{i'} \in G : p_{i'} \subseteq p_i \wedge s_i / \mathfrak{s}_{i'}$ .*

The first comparative case property describes consistency within a set of cases. A single case is consistent within a set of cases if and only if there does not exist any other case which solves the same or a more general problem differently. In general, this definition covers the issue of alternatives. We assume that for each possible problem there exists a "best" solution. For example, the quality of a solution is characterized by the complexity of its realization — is better to re-boot a computer than to buy a new one.

Consequently, we want the case base to consist only of cases that have "best" solutions and do not allow any alternative solutions. This also excludes the possibility of an alternative mentioned in the same case or as an extra case which has the same problem component but a different solution component. Although we do not want alternatives, we certainly must specify a concept, namely consistency, which enables the quality control mechanisms to detect alternatives and to suggest changes to the case base to avoid alternatives.

Moreover, definition 6 also covers situations where the subset relation between problem components with different solution components indicates that the more general problem definition possibly lacks some detailed information which is necessary to make the two cases distinctive.

**Definition 7 (Uniqueness).** Assume  $G \subseteq C$ , and  $c_i \in G$ .

$c_i$  unique within  $G : \Longleftrightarrow \nexists c_{i'} \in G, c_{i'} \neq c_i : p_{i'} = p_i \wedge s_i = s_{i'}$ .

Definition 7 declares a case as unique if and only if there does not exist another case within the considered set of cases which solves exactly the same problem in exactly the same way.

A more general relation which covers a similar situation is subsumption (see definition 8). A case subsumes another case if its problem component is a true subset of the problem component of the subsumed one and the solution component remains the same. We call cases without any different case that subsumes it minimal.

If we assume that the elements in the problem component define the problem as a conjunction of all elements, then the subsumes relation corresponds to solving a more general problem in the same way. The more specific problem characterization then possibly contains too many details unnecessary to specify the core problem. The restore step in CBM is responsible for resolving such situations accordingly.

**Definition 8 (Minimality).** Assume  $G \subseteq C$ , and  $c_i \in G$ .

$c_i$  minimal within  $G : \Longleftrightarrow \nexists c_{i'} \in G : p_{i'} \subsetneq p_i \wedge s_i = s_{i'}$ .

The most complex comparative case property describes situations with two cases within a set of cases that coincide in most of their components except for a specific number  $\Delta$  of values (see definition 9). We call a case incoherent within a set of cases if and only if there does not exist any other case which overlaps with the case in more than a pre-determined number of components. We consider incoherent cases as positive since the more cases differ, the broader is the spectrum of problems that they cover.

**Definition 9 (Incoherence).** Assume  $G \subseteq C$ ,  $c_i \in G$ , and  $1 \leq \Delta \in \mathbb{N}$ .

$c_i$  incoherent within  $G : \Longleftrightarrow \nexists c_{i'} \in G : |p_i \cap p_{i'}| + \Delta = N_i = N_{i'} \wedge s_i = s_{i'}$ .

With parameter  $\Delta$  we trigger the extent of overlapping information within coherent cases. For example, if  $\Delta$  is 1, two cases are coherent if all of their values are the same except for a single value (in each case). Note, this difference in a single component corresponds either to a situation with two different values for the same attribute or with different values for separate attributes.

For each pair of coherent cases, we possibly consider modification operators that generalize the two cases to a single one which represents both original cases. Removing the distinguishing values completely, joining them as a single set of alternatives, or generalizing to the next abstract level in a hierarchy of values are potential generalization operators.

*Example 2 (Case Properties).* Assume  $G$  in table 1, and  $\Delta = 1$ .

- (i)  $c_1$  is consistent within  $G$ ;  
 $c_4$  is not consistent within  $G$  except if  $s_2 \equiv s_3 \equiv s_4 \equiv s_5$ .
- (ii)  $c_1$  is unique within  $G$ ;  
 $c_2$  is not unique within  $G$  if  $s_2 \equiv s_3$ .
- (iii)  $c_1$  is minimal within  $G$ ;  
 $c_5$  is not minimal within  $G$  if  $s_2 \equiv s_5$  or  $s_3 \equiv s_5$ .
- (iv)  $c_2$  is incoherent within  $G$ ;  
 $c_6$  is not incoherent within  $G$  if  $s_1 \equiv s_6$ .

### 4.3 Case Base Properties

In the previous subsections, we defined several case properties to make decisions about the quality of cases. Now, we build upon these characteristics and specify properties for sets of cases, and hence for an entire case base which is simply a set of cases containing all cases in the case base.

**Definition 10 (Case Base Properties).** Assume  $G \subseteq C$ .

- (i)  $G$  is correct  $:\iff \forall c_i \in G : c_i$  correct.
- (ii)  $G$  is consistent  $:\iff \forall c_i \in G : c_i$  consistent within  $G$ .
- (iii)  $G$  is unique  $:\iff \forall c_i \in G : c_i$  unique within  $G$ .
- (iv)  $G$  is minimal  $:\iff \forall c_i \in G : c_i$  minimal within  $G$ .
- (v)  $G$  is incoherent  $:\iff \forall c_i \in G : c_i$  incoherent within  $G$ .

As case properties started with a notion of correctness for single cases, we also define a notion of correctness for a set of cases (see definition 10). This notion of correctness uses the correctness of single cases — a set of cases is correct if and only if all of its cases are correct.

In a similar vein, we also adopt the definition of consistent, unique, minimal, and incoherent. Definition 10 summarizes the extensions of these definitions for single cases to sets of cases in the same way as for correctness.

## 5 Initial Quality Measures for Case Base Maintenance

In this section, we use the previously defined case and case base properties to specify initial quality measures for case base maintenance. We start the set of initial quality measures with several degrees of quality. Definition 11 summarizes degrees of correctness, consistency, uniqueness, minimality, and incoherence.

**Definition 11 (Degrees of Case Base Properties).** Assume  $C^\subseteq$  is the set of all subsets of  $C$ .

$$\begin{aligned}
 (i) \quad \mathcal{D}_\surd : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{D}_\surd(G) := \frac{|\{c_i \in G \mid c_i \text{ correct}\}|}{|G|} \\
 (ii) \quad \mathcal{D}_\otimes : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{D}_\otimes(G) := \frac{|\{c_i \in G \mid c_i \text{ consistent within } G\}|}{|G|} \\
 (iii) \quad \mathcal{D}_{/\neq} : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{D}_{/\neq}(G) := \frac{|\{c_i \in G \mid c_i \text{ unique within } G\}|}{|G|} \\
 (iv) \quad \mathcal{D}_\sqsubset : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{D}_\sqsubset(G) := \frac{|\{c_i \in G \mid c_i \text{ minimal within } G\}|}{|G|} \\
 (v) \quad \mathcal{D}_\Delta : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{D}_\Delta(G) := \frac{|\{c_i \in G \mid c_i \text{ incoherent within } G\}|}{|G|}
 \end{aligned}$$

Each of these degrees computes the number of "good" cases in terms of the defined case properties and divides this number by the total number of cases within the considered set of cases. Hence, each degree counts the relative number of "good" cases within a given set of cases.

In comparison to the case base properties, these degrees enable a more refined consideration of the characteristics of the case base. The previously defined case base properties are essentially special cases for which each of the degrees yields value 1.

The purpose of the different degrees in terms of the maintenance steps in CBR is to get values that provide an indicator for the review step and the decision whether the case base state is still acceptable or not. For example, we specify a threshold and then classify a case base as acceptable if and only if each of the degrees is above this threshold.

Of course, we also think of more complex monitor operators that consider the degree values over time and react on the dynamic changes of the resulting functions. For example, we invoke maintenance in terms of applying modify operators as soon as the difference between two measurements is too large into a decreasing direction rather than only comparing absolute degree values and a pre-specified threshold.

**Definition 12 (Quality Measures).** Assume  $C^\subseteq$  is the set of all subsets of  $C$ , and  $w_\surd, w_\otimes, w_{/\neq}, w_\sqsubset, w_\Delta \in [0; 1]$  with  $\sum_{x \in \{\surd, \otimes, /\neq, \sqsubset, \Delta\}} w_x = 1$ .

$$\begin{aligned}
 (i) \quad \mathcal{Q}_{\min} : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{Q}_{\min}(G) := \min_{x \in \{\surd, \otimes, /\neq, \sqsubset, \Delta\}} \{\mathcal{D}_x(G)\} \\
 (ii) \quad \mathcal{Q}_{\max} : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{Q}_{\max}(G) := \max_{x \in \{\surd, \otimes, /\neq, \sqsubset, \Delta\}} \{\mathcal{D}_x(G)\} \\
 (iii) \quad \mathcal{Q}_w : C^\subseteq &\rightarrow [0; 1], \quad \mathcal{Q}_w(G) := \sum_{x \in \{\surd, \otimes, /\neq, \sqsubset, \Delta\}} w_x \cdot \mathcal{D}_x(G)
 \end{aligned}$$

Definition 12 shows three examples for quality measures in CBR that cumulate the various degrees specified above to a single overall value of case base quality. The first variant considers the minimum value of degrees as the crucial number for quality control. If we assume that we compare quality values with a given



threshold to trigger the restore step, this means that quality control based on this measure invokes modify operators as soon as a single degree — regardless which one — becomes inappropriate. The second example in definition 12 does exactly the opposite. It starts to recommend changes to the case base only if all degrees depict the same bad quality.

The third measure is a compromise between those two extremes and considers the average degree value if we set all weights to the same value. On the other hand, this measure also allows the user to set preferences on specific aspects. For example, the user does not care about redundant cases and sets  $w_{/=}$  to zero, but does not want any inconsistencies within any subset of cases in the case base and consequently initializes  $w_{\otimes}$  to a relatively high value.

*Example 3 (Quality Measures).* Assume  $G$  in table 1,  $G$  is correct,  $s_1 \equiv s_6$ ,  $s_2 \equiv s_3 \equiv s_5$ ,  $\Delta = 1$ , and  $w_{\surd} = w_{\otimes} = w_{/=} = w_{\subseteq} = w_{\Delta} := 1/5$ .

Then,  $\mathcal{D}_{\surd}(G) = 1$ ,  $\mathcal{D}_{\otimes}(G) = 5/6$ ,  $\mathcal{D}_{/=}(G) = 2/3$ ,  $\mathcal{D}_{\subseteq}(G) = 5/6$ , and  $\mathcal{D}_{\Delta}(G) = 2/3$ . Moreover,  $\mathcal{Q}_{min}(G) = 2/3$ ,  $\mathcal{Q}_{max}(G) = 1$ , and  $\mathcal{Q}_w(G) = 4/5$ .

We get more alternative quality measures if we allow the weights  $w_x$  to be any function of  $G$ , or by adding aspects of time, or by taking into account measures of usage information, and so on. At this point, the concept of specifying quality measures is sufficiently general to enable appropriate definitions of respective quality control mechanisms in almost any domain.

## 6 Concluding Remarks

In summary, we considered case base maintenance as one of the most important issues in current CBR research and suggested two novel steps, the review and the restore steps, within the maintenance phase of the CBR process. We emphasized quality control in the review step and proposed several quality measures based on case and case base properties to monitor the quality of the case base.

Beyond the exact and precise definition of a framework which covers all aspects and tasks in CBM, our suggestions for future work point in several directions. First, we further analyze the specified quality measures, their relation to each other, and how their values actually affect the case base. Second, we plan to implement the quality measures and to test if they are really appropriate to measure the quality of a case base and to indicate when the restore step in maintenance is necessary. This issue implies that we also define different mechanisms to monitor the quality values and to specify concrete moments to invoke the restore step.

The current definition of quality measures considers local similarity only as a matching function — for this aspect, we also plan to extend our definitions to other more fine-grained local similarity measures. Moreover, extensions to other knowledge containers beyond the case base are an important issue for future work.

Another line of future research is the restore step itself. We have to define which modify operators are necessary to change the case base and to improve

its current quality. In particular, this task also means that we have to analyze dependencies between quality control and modification of the case base.

Finally, issues of computation play an additional role. For example, if the case base becomes too large to compute the quality measures in a reasonable amount of time, we have to consider aspects of clustering cases in order to compute quality values in local case base areas and then cumulate these local values to an overall quality of the entire case base.

## Acknowledgments

We thank Andre Hüttemeister and Mehmet Göker for their initial ideas on quality measures for maintenance in CBR which led to the definitions presented in this paper.

## References

1. Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994. 248
2. David W. Aha and Leonard A. Breslow. Refining conversational case libraries. In *Proceedings of the Second International Conference on Case-Based Reasoning*, pages 267–278, 1997. 250
3. Mehmet Göker and Thomas Roth-Berghofer. The development and utilization of the case-based help-desk support system HOMER. *Special Issue of the International Journal “Engineering Applications of Artificial Intelligence”*, 12(6), 1999. 248
4. David B. Leake and David C. Wilson. Categorizing case-base maintenance: Dimensions and directions. In *Proceedings of EWCBR-98, Advances in Case-Based Reasoning*. Springer-Verlag, 1998. 247
5. David B. Leake and David C. Wilson. When experience is wrong: Examining CBR for changing tasks and environments. In *Proceedings of the Third International Conference on Case-Based Reasoning*, 1999. 250
6. Kirsti Racine and Qiang Yang. On the consistency management of large case bases: the case for validation. In *Proceedings of the AAAI-96 Workshop on Knowledge Base Validation, American Association for Artificial Intelligence (AAAI)*, 1996. 250
7. Kirsti Racine and Qiang Yang. Maintaining unstructured case bases. In *Proceedings of the 1997 International Conference on Case Based Reasoning*, pages 553–564, 1997. 250
8. Michael M. Richter. The knowledge contained in similarity measures. Invited Talk at the International Conference on Case-Based Reasoning, 1995. 247
9. Barry Smyth and Mark T. Keane. Remembering to forget: A competence-preserving deletion policy for case-based reasoning systems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 377–382, 1995. 250
10. Barry Smyth and Elizabeth McKenna. A portrait of case competence: Modelling the competence of case-based reasoning systems. In *Proceedings of the 4th European Workshop on Case-Based Reasoning*., pages 208–220, 1998. 250

11. Jun Zhu and Qiang Yang. Remembering to add: Competence-preserving case addition policies for case base maintenance. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, 1999. 250

# A New Approach for the Incremental Development of Adaptation Functions for CBR

Abdus Salam Khan and Achim Hoffmann

Artificial Intelligence Laboratory  
School of Computer Science and Engineering  
The University of New South Wales, Sydney 2052, Australia  
{askhan,achim}@cse.unsw.edu.au

**Abstract.** This paper introduces a new approach to building complex adaptation functions for case-based reasoning systems.

We present an incremental method which allows a domain expert to refine the existing adaptation function during use of the system. We lend ideas from Ripple-Down Rules, a proven method for the very effective and efficient acquisition of classification knowledge during the use of a knowledge-based system. In our approach the expert is only required to provide explanations of why, for a given problem, a certain adaptation step should be taken. Incrementally a complex adaptation function as a systematic composition of many simple adaptation functions is developed. This approach is effective with respect to both, the development of highly tailored and complex adaptation functions for CBR as well as the provision of an intuitive and feasible approach for the expert.

The approach has been implemented in our CBR system MIKAS, for the design of menus according to dietary requirements.

While our approach showed very good results in MIKAS, it represents also a promising technique for many other CBR applications.

## 1 Introduction

Practical experiences in CBR have over and over demonstrated the difficulty of developing suitable adaptation functions for a CBR system [8,1,4]. It is clear that the adaptation function is highly dependent on the domain and hence has to be provided either by the domain expert himself/herself or the system developer has to closely collaborate with a domain expert. See, e.g., [8,11] for the difficulties in obtaining adaptation knowledge.

Hence, it is difficult to provide all the required knowledge to master these difficulties [10]. It is also difficult to design a suitable adaptation function. The problems are so acute that many CBR applications simply omit case adaptation [7].

Experiences from knowledge acquisition for expert systems have also shown that it is usually very difficult to obtain the relevant knowledge from an expert, as experts are usually unable to provide precise rules which would describe their decisions.

What experts can usually do, is to explain why they took a certain decision in a given situation. Providing general rules when to take what decision is much harder as it would require the expert to consider all sorts of situations which would be covered under the conditions he/she mentions in the rules.

Ripple-Down Rules have been developed as an extremely effective approach for the acquisition of classification knowledge, as they require the expert only to provide explanations of the decision taken in a given situation.

In this paper, we introduce a radically new approach for developing a suitable adaptation function. We encode adaptation knowledge in rules, which somewhat resemble the rules used in the INRECA approach [9,1]. The experiences in the INRECA project also showed the difficulty of actually encoding the suitable adaptation knowledge [1]. We address this experienced difficulty with our new approach: a new way, how an expert interacts with the system in order to provide suitable adaptation rules. Our new approach for the *acquisition of adaptation knowledge* is based on ideas of Ripple-Down Rules [2].

The paper is organised as follows: In the next section, we illustrate to challenges of our domain of menu design for which we developed a CBR system using our new approach. Section 3 presents the technical details of our approach. Section 4 illustrates, how the approach is implemented in our menu design system MIKAS. The conclusions are found in section 5.

## 2 The Challenges in Menu Design

To illustrate the requirements for our CBR system MIKAS, we give a brief explanation of what kind of problems the system has to deal with in this domain.

A menu is recommended by a dietitian for a patient with specific health conditions. These health conditions represent stringent requirements on the menus to be designed. This does not only include certain amounts of various nutrients which should be contained in the diet. It may also refer to certain ways of preparing foods, e.g., use of spices, certain types of foods (vegetarian, kosher meat, etc.), or certain other aspects of the involved foods, such as their texture or flavour. The significance of this is that for 'normal' menus it seems appropriate to retrieve cases according to how well they match the required nutrients. However, adaptation may be necessary in order to scale the amount of foods to match nutrient requirements or to remove or replace certain types of foods which are unsuitable for the given patient.

We give a few examples for illustrating possible requirements:

1. Patient: Anaemic, diabetic and has high blood pressure. Furthermore, the patient is allergic to garlic.

Diet requirements: The iron content should be fairly high due to anaemia, sugar content should be very low due to diabetes, fat, cholesterol and cooking salt (sodium) should be minimised due to high blood pressure and the menu must not contain any food that has garlic in the preparation due to the patient's allergy.

2. Patient: Anaemic, diabetic and has high blood pressure. Furthermore, the patient is allergic to garlic. In addition, the patient has duodenal ulcer.

Diet requirements:

Same requirements as for the diet in patient 1.

In addition, due to duodenal ulcer, additional snacks in regular intervals are recommended to minimise damage of gastric lining. Bedtime snacks should be avoided as they will cause nocturnal acid secretion. Black pepper and chili powder should be avoided because they may cause distress. Caffeine should be avoided as it increases gastric secretion.

3. Patient: Kidney disease with glomerular filtration rate (GFR) less than 20-25 ml/minute and plasma urea level greater than 20mmol/L.

Diet requirements: Low protein content in order to lessen the workload for kidneys, (protein rich food can increase levels of waste products leading to symptoms such as nausea, loss of appetite, vomiting and tiredness). Sodium content should be kept low to prevent fluid retention. Foods high in sodium are avoided because salty foods can cause fluid retention and can result in high blood pressure, oedema and shortness of breath. Potassium should be kept low to prevent abnormal cardiac rhythms and contractions. Potassium restriction limits foods like milk, meat, whole-grain products, most fruit and vegetables. No salt should be used in cooking or added at the table.

Patient 1 and 2 have similar conditions. So it appears appropriate to retrieve a menu for patient 2 which was already used for patient 1. However, the additional requirements would need to be accommodated by adaptation. If a bedtime snack is removed, some other foods may need to be increased in order to match the nutrient requirements. If black pepper should be avoided, a main meal may need to be replaced entirely, as the peppered meat portion in the retrieved menu may not be acceptable.

If a menu is retrieved for patient 3, which was used for patient 1 or 2, again adaptation steps need to be taken, which would reduce the salt content. That is, all foods which are rich in salt would need to be replaced or deleted from the menu. If there are many such foods, a different menu should be retrieved, otherwise adaptation may be unable to handle the problem.

### 3 Incremental Development of the CBR Adaptation Function

In this section, we present our approach to incrementally develop highly specialised and complex adaptation functions for CBR systems by direct interaction of the expert with the system.

#### 3.1 Motivation for Our Approach

Building Case-Based Reasoning systems successfully often requires the development of a highly specialised retrieval function, which is complemented by a

specialised adaptation function tailored to the domain. It must also be ensured that the retrieved case can be successfully adapted if required. While adaptability may seem warranted, if the retrieved case is reasonably similar to the problem specification, this may not be so, as the adaptation module may not be able to bridge all kinds of differences equally well.

For example, in our system MIKAS for designing menus according to dietary requirements, cases are menus along with patient descriptions.

A suitable menu has not only to match the nutritional requirements but also a large number of special requirements for various health conditions need to be accommodated. Such special requirements may range from avoiding spicy food, over minimising the salt content, to ensuring that all foods come in sufficiently small parts which may even exclude thick drinks, such as milk shakes, due to the patient's difficulties when swallowing.

We lend ideas from Ripple-Down Rules [2], that ensures an incremental coverage of the expert's judgements and decision making. That is, for all problem cases for which the expert provided advice to the system on how to adapt a retrieved case, the system will have to reproduce the expert's performance.

We assume a domain expert will have a sufficient understanding of how to adapt a retrieved case in order to find a solution to the current problem.

### 3.2 Ripple-Down Rules

The basic idea of Ripple-Down Rules (RDR) [2,3,5] is to develop a knowledge base by allowing the expert to directly interact with the system and to incrementally add rules to a knowledge base.

In traditional rule-based expert systems, the expert (in conjunction with a knowledge engineer) is required to specify a number of abstract rules which, by allowing complex inferences, hopefully cover the entire domain correctly. While this is known to be usually a very difficult and time-consuming task, RDR make the job of the expert much easier:

In RDR, the object space to be classified is incrementally subdivided into smaller and smaller partitions, until all objects in each single partition belong to the same class. The rules for subdividing the object space are specified by the expert, whenever an object is encountered, which the system classifies in disagreement with the expert. That is, the current object  $x$  falls into a partition  $p$  of the object space, which classifies the object incorrectly. Hence, this partition needs to be further subdivided into two partitions  $p_1, p_2$ , such that the partition to which  $x$  belongs, classifies  $x$  correctly. Such a subdivision can be provided by the expert competently and with minimal effort, as the expert is merely required to provide an explanation of why  $x$  is different from the previously presented object  $x_p$ , which led to the creation of the partition  $p$  in the first place. That is, the expert needs only to provide a criterion by which  $x$  differs from  $x_p$  and which justifies the different classification. To provide such an explanation is usually easy for an expert as it is not much different to explaining their decision to colleagues.

See figure 1 for an RDR tree. An object is classified by this tree as follows: Initially the 'default rule' in node 1 is evaluated, i.e. class '-' is obtained. Before

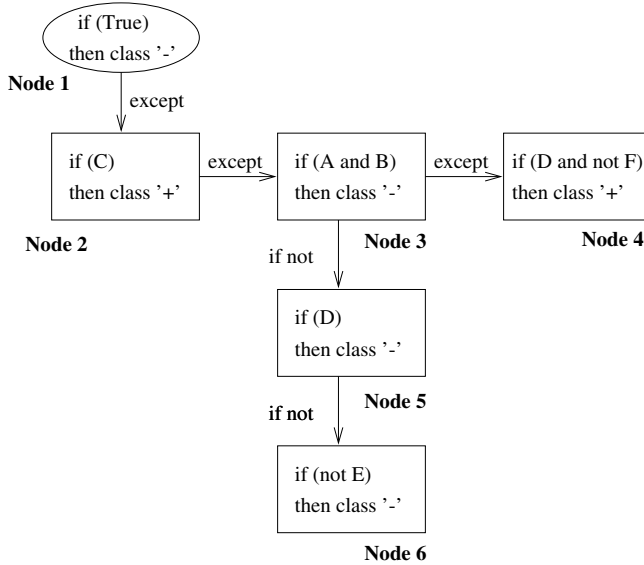


**Fig. 1.** An example of an Ripple Down Rules tree

this becomes the final 'verdict', it is checked whether any 'except' link from the current node exists. If there is such a link, the connected node's condition is checked.

Here, we check the condition of node 2: if "C" is true, then the corresponding class '+' overwrites the previous 'verdict', unless this in turn is overwritten by another except link. If an except link exists, but the connected rule condition is not satisfied, say in node 3, then the nodes along a possibly existing chain of 'if not' links are checked. For instance, if the condition of 2 is satisfied, node 3 is checked. If the condition of node 3 is not satisfied, then node 5 is checked and, if node 5 is not satisfied, node 6 is checked. If any of these nodes 3, 5, or 6 is satisfied, the classification of the first satisfied node is the final verdict, unless this node has again an overwriting except link.

If the expert is not satisfied, say with the classification due to node 5, the system asks for a justification in terms of the current object's attributes. This explanation is then used as the condition to a new exception rule to node 5.

### 3.3 Formal Preliminaries to Our New Approach to Acquire Adaptation Knowledge

In general we consider attributes to be either numerical or discrete valued.<sup>1</sup>

<sup>1</sup> Numerical valued attributes may be integer or continuously valued - the critical point is that the possible values have a meaningful order, while the discrete valued attributes have a finite value range without any meaningful order.



We describe a case  $C = \{\mathcal{P}, \mathcal{S}\}$  by two attribute vectors with the domains  $P_1 \times P_2 \times \dots \times P_n$  and  $S_1 \times S_2 \times \dots \times S_m$  respectively.

- The attribute vector  $P = \langle p_1, \dots, p_n \rangle$  represents the problem specification.
- The attribute vector  $S = \langle s_1, \dots, s_m \rangle$  is an ordered list of components and represents a solution to the problem  $P$ .

We call *Comps* the *available component list*. This is the general list of all possible types of components which can be included in the design. Hence, the components in the solution are also taken from the general list of available components, i.e.  $s, \dots, s_m \in \text{Comps}$ .

Each component  $c \in \text{Comps}$  is described by a number of attributes, called the *component attributes*, i.e.  $c = \langle c_{a,1}, c_{a,2}, \dots, c_{a,k} \rangle$ . The domain of each component attribute  $c_{a,i}$  is denoted by  $C_{a,i}$ . That is, each component is an element of  $C_{a,1} \times \dots \times C_{a,k}$ .

In our case of diet construction, this is a database of possible foods, describing the nutrient content and food type of each food.

The case base contains cases, which are composed of a problem specification and a solution. The solution is a design, composed of a number of components which are implicitly related to each other by their respective position in the ordered list of components  $S$ .

Generally speaking, the problem statement is composed of two parts:

- an attribute-value vector, stating certain properties of the problem, in order to choose the solution from an appropriate category.
- a collection of constraints, which either require certain values for certain slots in a solution and/or which forbid certain values for certain slots.

Constraints may also specify that certain combinations of values of multiple attributes should or should not occur in a solution.

In the particular case of diet construction, our components are the various foods we have available and they are related to each other by composing the various meals of the day.

A number of constraints on the possible designs will apply in general, while other constraints apply only to solutions for a specific class of problems. In diet construction, there are general constraints such as the required energy and nutrient level of the diet as well as specific constraints such as to choose an appropriate diet for vegetarians or diabetes patients, etc.

Some constraints are explicitly stated in the problem description, e.g., the amounts of the required daily intakes of nutrients for the task of diet construction. Other constraints are not explicitly stated but must be provided by a domain expert. For diet construction, this includes constraints on which types of foods to use for a specific meal to ensure that it represents a sensible composition of foods, such as meat combined with potatoes or rice combined with vegetables or salad.

### 3.4 The Knowledge Acquisition Process for Case Adaptation

After cases have been retrieved, the system tries to adapt the few highest ranked cases in order to fit the current problem.

If the system cannot produce a satisfactory solution, additional adaptation knowledge must be provided to the system. Generally speaking, this can either be a refined retrieval function, or a refinement of the adaptation function. The expert has to decide whether the retrieved cases are feasible to be adapted or whether other cases need to be retrieved.

In the following, we discuss what happens, if the expert decides that the retrieved cases should be adapted, i.e. if the expert decides to enhance the system's adaptation function.

Firstly, the expert tries to manually adapt the solution of the retrieved cases so that a solution for the current problem is found. This adaptation is done by removing and adding components.

After the expert found a satisfactory adaptation of one of the retrieved solutions, the system requests explanations for each adaptation action the expert chose. To do so, the expert has to provide a condition under which the chosen adaptation action (the removal or addition of a component) should be executed. This condition is expressed, generally speaking, in terms of the current problem description as well as the retrieved solution to be adapted.

Furthermore, the action needs to be specified beyond being either a deletion or an addition of a component: The particular component which is added or deleted from the particular slot of the solution needs to be specified. This is done again by providing conditions which a component has to meet to be eligible for deletion or addition from or to a particular slot or class of slots. The new rule  $r'$  is integrated into the existing Ripple Down Rules structure as an exception to the rule  $r$ , which produced the undesired adaptation action, which the expert intends to supersede by the new rule. The system ensures that the expert entered conditions, which do not apply to all those cases, to which the previous rule  $r$  applied successfully, i.e. for which rule  $r$  led to an adaptation action, which was accepted by the expert.

**Abstract Actions** The purpose of *abstract actions* is to allow a way of generalisation. An adaptation rule is provided by an expert, who encounters an individual case which needs to be adapted, while observing the system's performance. The expert will be able to decide which component should be replaced by which new component.

However, just to provide the identity of those components will render the CBR system incapable to adapt a case which contains not exactly the same but perhaps a very similar component. Similarly, the new component may need some variation in a new case which need to be adapted. As a consequence, our system lets the expert specify the action, he/she suggests for adaptation of the given case, and then asks the expert to abstract from the individual action and to give a more abstract description of that action (usually a replacement of a component, an increase or decrease in the amount or size of a component).

An abstract action can be considered as a set of rules in itself. That is, depending on the features of the solution to be adapted, the abstract action may result in different changes to a different case. One important aspect is, that numerical feature values can be used to determine numerical aspects of the change. Most notably, the numerical attributes of components which are to be removed or added to the design can be determined by largely simple arithmetic calculations.

Formally, we allow to define abstract actions by a set of rules. Each rule has a condition part as described in the previous subsection and an action part. The action to be specified is either to delete, to add a certain component or to increase or decrease the amount/size of a component, which can be identified using certain attributes of the component and whose numerical attributes can be described using the attribute values of any part of the current case. A *replacement-action* is then composed of a *delete-action* and an *add-action*.

A *delete-action* just needs to identify the component to be deleted. This is done by specifying attribute value ranges for the attributes of components, which have to be matched by the component which is deleted from the current case.

The formal description is essentially the same as for the conditions of adaptation rules as mentioned earlier, although the involved attributes do not describe the case but rather the component.

The first component  $c = \langle c_{a,1}, \dots, c_{a,n} \rangle \in Comps$  will be deleted, which is found in the component list of the case and for which the following condition holds:

$$(c_{a,i_1,min} \leq c_{a,i_1} \leq c_{a,i_1,max} \wedge \dots \wedge c_{i_k,min} \leq c_{a,i_k} \leq c_{i_k,max}),$$

where  $c_{a,i_1}, \dots, c_{a,i_k} \in \{c_{a,1}, \dots, c_{a,n}\}$ .

Similarly, *add-actions* can be defined as follows:

Add the component  $c = \langle c_{a,1}, \dots, c_{a,n} \rangle$  from the *available component list*, for which the following condition holds:

$$(c_{a,j_1,min} \leq c_{a,j_1} \leq c_{a,j_1,max} \wedge \dots \wedge c_{j_k,min} \leq c_{a,j_k} \leq c_{j_k,max}),$$

where  $c_{a,j_1}, \dots, c_{a,j_k} \in \{c_{a,1}, \dots, c_{a,n}\}$ .

## 4 Acquiring Adaptation Knowledge with MIKAS

MIKAS implements the above sketched strategy for developing a CBR system. We used our RDR based approach for both phases, case retrieval [6] and adaptation. In this paper, we will only present the adaptation process.

During a diet planning session a dietitian (the expert) usually discusses with the patient (client) the history of the patient's diet. The expert then analyses the intake (nutrients of consumed food), compares that intake with the recommended dietary intake (RDI) and finally designs a diet for the patient.

Our system MIKAS accepts a description of the patient along with nutritional requirements. The health condition of the patient would imply certain

restrictions on the food which can be used in a diet. These restrictions are not explicitly given, but acquired, largely in the form of adaptation rules, provided by the expert.

For example, the diet may contain some foods which should not be recommended for the patient because of some possible side-effects, e.g., spicy foods are harmful for a patient having duodenal ulcer. The expert provides the overall adaptation function incrementally, while using the system. For every newly encountered problem, the expert may add another rule to the RDR structure, if the existing RDR structure does not perform satisfactorily.

After the problem description has been entered, the expert considers the retrieved menus. If he is not satisfied with the retrieval result, the retrieval function will be enhanced. Otherwise, he tries to adapt one of the retrieved menus, using adaptation steps from the selection menu shown in figure 2 (left).

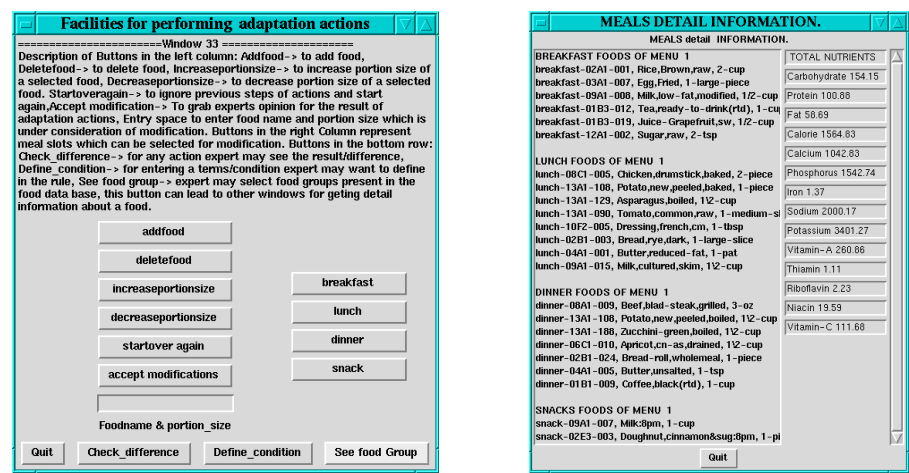


**Fig. 2.** Left: Window for choosing an adaptation action in MIKAS. Right: A retrieved menu, which is to be adapted by the expert. The expert provides adaptation rules as explained in the text

### A Sketch of a Knowledge Acquisition Session with MIKAS

The patient description of a 60-years old patient, who usually has non-vegetarian, English meals and who has duodenal ulcer was entered into MIKAS. The system retrieved menu 1 as shown in figure 2 (right). The expert did not agree with the retrieved menu for various reasons, including the coffee in the dinner, which is harmful for ulcer patients, etc.

So the expert modified the retrieved diet by taking out the breakfast grapefruit juice which is considered as harmful for the patient. The expert further

increased the portion size of the asparagus in the lunch in order to better match the nutrient requirements. He further deleted the dinner coffee for its caffeine content, and added baked custard to the snack slot for 11am, as an ulcer patient needs frequent meals in regular intervals. At the end of the adaptation steps, the expert found the resulting menu suitable for the patient.

After that, MIKAS asked the expert for justifications for each of the actions taken. That is, the expert was asked to provide conditions, which have to match the current case, under which the corresponding action should be taken.

Furthermore, MIKAS asks the expert, to describe to taken action in more abstract terms. That is, if the expert deleted the grapefruit juice from the breakfast slot, the expert was asked whether only grapefruit juice or any fruit juice or even any fruit meeting specifiable characteristics, should be deleted. Furthermore, whether the action should only be applied to the breakfast or to other menu sections as well.

The previously entered adaptation steps were the following:

- Delete grapefruit juice from breakfast
- Increase portion size of asparagus from lunch slot
- Delete coffee from the dinner slot
- Add baked custard to the snack slot.

In another session a description of a patient, who is much younger in age but has similar health conditions was entered into MIKAS, together with nutrient requirements. The nutrient requirements for this patient were higher, because he is younger and, hence, needs more energy supply and more protein as he is more active. For this patient, MIKAS retrieved menu 2 in figure 3 (left). The rules entered before and applied to menu 2 resulted in the following actions:

1. there was no grapefruit juice in the breakfast slot, but MIKAS deleted orange&mango juice instead. Reason being that it belonged to the same group of fruit juices, as specified in the abstract action by the expert.
2. MIKAS did not find asparagus in the lunch slot of menu 2, so baby carrots were taken instead and the portion size was increased.
3. MIKAS deleted coffee from the dinner slot.
4. MIKAS added baked custard to the snack slot.

If the expert finds the deletion of orange&mango juice inappropriate in the case of menu 2, he can say so and suggest an alternative action. MIKAS will then ask again for conditions under which the alternative action should be taken. These conditions must differentiate between menu 1 and menu 2 and/or between the first and second patient, so that the first rule would still apply in the first case and the new rule would only apply to menu 2.

This is accommodated in the structure of the Ripple-Down Rules and ensures that the task to provide suitable conditions can be easily accomplished by a domain expert.

| MEALS DETAIL INFORMATION.                         |                     |
|---------------------------------------------------|---------------------|
| MEALS detail INFORMATION.                         |                     |
| BREAKFAST FOODS OF MENU 2                         | TOTAL NUTRIENTS     |
| breakfast-01B3-008, Juice,orange& mango, 1-cup    | Carbohydrate 169.59 |
| breakfast-02D1-005, Rice-bubbles, 1-cup           | Protein 126.63      |
| breakfast-03A1-004, Milk,skim,fluid, 1-cup        | Fat 30.77           |
| breakfast-03A1-008, Egg,poached, 1-large-piece    | Calorie 1470.94     |
| breakfast-01B1-012, Tea,herbal(rtd), 1-cup        | Calcium 914.04      |
|                                                   | Phosphorus 1751.28  |
| LUNCH FOODS OF MENU 2                             | Iron 1.23           |
| lunch-08A2-005, Lamb,chump,grilled, 3-oz          | Sodium 2184.7       |
| lunch-02A1-004, Rice,sungold,boiled, 1-cup        | Potassium 2369.87   |
| lunch-13A1-145, Carrot,baby,boiled, 1/2-cup       | Vitamin-A 259.5     |
| lunch-13A1-090, Tomato,common,raw, 3-thin-slice   | Thiamin 1.02        |
| lunch-13A1-053, Lettuce,common,raw, 1/2-cup       | Riboflavin 1.7      |
| lunch-09D2-004, Custard,baked, 1/2-cup            | Niacin 21.57        |
| lunch-01B1-014, Coffee,decaf,black(rtd), 1-cup    | Vitamin-C 196.49    |
| DINNER FOODS OF MENU 2                            |                     |
| dinner-05A1-029, fish,Whiting,steamed, 2-fillet   |                     |
| dinner-13A1-108, Potato-peeled,boiled, 1-piece    |                     |
| dinner-13A1-206, Bamboo,shoot,cn, 1-cup           |                     |
| dinner-02B1-008, Bread,wholemeal, 1-large-slice   |                     |
| dinner-10C1-002, Soup,tomato,cn,prep, 1-cup       |                     |
| dinner-09D2-004, Ice-cream,super-premium, 1-cup   |                     |
| dinner-01B1-009, Coffee,black(rtd), 1-cup         |                     |
| SNACKS FOODS OF MENU 2                            |                     |
| snack-09B2-005, Cheese,cottage,low-fat,3pm, 1/2-c |                     |
| snack-20C1-002, Cracker,flaky, 8-piece:3pm        |                     |
| Quit                                              |                     |

| MEALS DETAIL INFORMATION.                         |                      |
|---------------------------------------------------|----------------------|
| MEALS detail INFORMATION.                         |                      |
| BREAKFAST FOODS OF MENU 2                         | TOTAL NUTRIENTS      |
| breakfast-02D1-005, Rice-bubbles, 1-cup           | Carbohydrate 187.142 |
| breakfast-03A1-004, Milk,skim,fluid, 1-cup        | Protein 140.228      |
| breakfast-03A1-008, Egg,poached, 1-large-piece    | Fat 42.708           |
| breakfast-01B1-012, Tea,herbal(rtd), 1-cup        | Calorie 1705.04      |
|                                                   | Calcium 1238.86      |
| LUNCH FOODS OF MENU 2                             | Phosphorus 2042      |
| lunch-08A2-005, Lamb,chump,grilled, 3-oz          | Iron 2.744           |
| lunch-02A1-004, Rice,sungold,boiled, 1-cup        | Sodium 2389.16       |
| lunch-13A1-145, Carrot,baby,boiled, 2-cup         | Potassium 2575.61    |
| lunch-13A1-090, Tomato,common,raw, 3-thin-slice   | Vitamin-A 416.44     |
| lunch-13A1-053, Lettuce,common,raw, 1/2-cup       | Thiamin 1.138        |
| lunch-09D2-004, Custard,baked, 1/2-cup            | Riboflavin 2.392     |
| lunch-01B1-014, Coffee,decaf,black(rtd), 1-cup    | Niacin 21.699        |
| DINNER FOODS OF MENU 2                            | Vitamin-C 119.47     |
| dinner-05A1-029, fish,Whiting,steamed, 2-fillet   |                      |
| dinner-13A1-108, Potato-peeled,boiled, 1-piece    |                      |
| dinner-13A1-206, Bamboo,shoot,cn, 1-cup           |                      |
| dinner-02B1-008, Bread,wholemeal, 1-large-slice   |                      |
| dinner-10C1-002, Soup,tomato,cn,prep, 1-cup       |                      |
| dinner-09D2-004, Ice-cream,super-premium, 1-cup   |                      |
| SNACKS FOODS OF MENU 2                            |                      |
| snack-09B2-005, Cheese,cottage,low-fat,3pm, 1/2-c |                      |
| snack-20C1-002, Cracker,flaky, 8-piece:3pm        |                      |
| snack-09D2-004, Custard,baked 1/2-cup             |                      |
| Quit                                              |                      |

**Fig. 3.** A new menu before (left) and after (right) adaptation using the rules entered while adapting menu 1

## 5 Conclusion

We introduced a new approach to the incremental development of complex adaptation functions for Case-Based Reasoning systems. The approach is based on Ripple-Down Rules [2], which has been proved to be an excellent technique for the effective acquisition of classification knowledge directly from experts.

The strengths of our approach include the following:

- it allows a domain expert to directly interact with the system without the need for a knowledge engineer or a system engineer. Adaptation rules are entered by the expert. The expert is guided by the system in providing suitable conditions for the rule's application.
- the expert is only required to explain, why certain solutions should be adapted in the demonstrated way for a given problem. This is much easier than to provide general rules for adaptation. In our approach, the expert needs only to justify the particular adaptation steps he chose.
- If an explanation results in overly general adaptation rules, this will be repaired as soon as the expert encounters a situation, in which the system would perform unsuitable adaptations.

In such a situation, the expert simply tells the system, what instead should be done and provides an explanation for it. The explanation is automatically converted into a new adaptation rule. The new rule  $r'$  is integrated into the entire set of adaptation rules to ensure that it will be used in place of the previous rule  $r$  only in the encountered situations as well as in those related situations as specified by the expert's explanation. In cases where the old

rule  $r$  has been applied successfully (as accepted by the expert) it is ensured that  $r$  still applies and is not superseded by  $r'$ .

- the approach can be carried over to the retrieval phase in CBR as well, as described in [6].

Our approach allows a domain expert, to directly develop a complex adaptation function without the assistance of a systems engineer. The expert is only required to adapt one of the solutions of the retrieved cases to generate a suitable solution for the problem at hand. Once such a suitable adaptation is found, the system requests from the expert explanations for why the chosen steps should be taken. From the expert's explanations, the system refines its adaptation knowledge by extending the Ripple-Down Rule tree which controls the adaptation steps to be taken.

Due to the structure of Ripple-Down Rule trees, those refinements do not interfere with previous refinements, as the applicability of the new refinement is restricted to problems sufficiently related to the new problem at hand.

## References

1. R. Bergmann, S. Breen, E. Fayol, M. Goker, M. Manago, S. Schmitt, J. Schumacher, A. Stahl, S. Wess, and W. Wilke. Collecting experience on the systematic development of CBR applications using the INRECA methodology. In *Proceedings of the European Workshop on Case-Based Reasoning, EWCBR98*, 1998. 260, 261
2. Paul Compton, Glen Edwards, Byeong Kang, Leslie Lazarus, Ron Malor, Phil Preston, and Ashwin Srinivasan. Ripple down rules: Turning knowledge acquisition into knowledge maintenance. *Artificial Intelligence in Medicine*, 4:463–475, 1992. 261, 263, 270
3. Paul Compton, Byeong Kang, Phil Preston, and Mary Mulholland. Knowledge acquisition without knowledge analysis. In *Proceedings of the European Knowledge Acquisition Workshop*, pages 277–299. Springer-Verlag, 1993. 263
4. K. Hanney and M. T. Keane. When do you need adaptation?: A review of current practice. In *AAAI-95 Fall Symposium on Adaptation in Knowledge Reuse, Boston, U.S.A.*, 1995. 260
5. Byeong Kang and Paul Compton. A maintenance approach to case-based reasoning. In J. P. Haton, K. Keane, and M. Manago, editors, *Advances in Case-Based Reasoning*, pages 226–239. Springer-Verlag, 1995. 263
6. A. S. Khan and A. Hoffmann. Incremental acquisition of retrieval knowledge for case-based reasoning. In *Proceedings of the Fourth Australian Knowledge Acquisition Workshop*, pages 72–86, Sydney, Australia, December 5-6 1999. University of New South Wales. 267, 271
7. D. B. Leake. Combining rules and cases to learn case adaptation. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science*, pages 84–89, 1995. 260
8. D. B. Leake. *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. Menlo Park: AAAI Press, 1996. 260
9. W. Wilke and R. Bergmann. Techniques and knowledge used for adaptation during case-based problem solving. In *(AIPS'98) Workshop on Integrating Planning, Scheduling and Execution in dynamic and uncertain Environment*. AAAI press, 1996. 261

10. W. Wilke and R. Bergmann. Techniques and knowledge used for adaptation during case-based problem solving. In *11<sup>th</sup> International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert System*, 1998. 260
11. W. Wilke, I. Vollrath, K. Althoff, and R. Bergmann. A framework for learning adaptation knowledge based on knowledge light approaches. In *5<sup>th</sup> German Workshop on Case-Based Reasoning (GWCBR'97)*, 1997. 260



# An Efficient Approach to Similarity-Based Retrieval on Top of Relational Databases

Jürgen Schumacher<sup>1</sup> and Ralph Bergmann<sup>2</sup>

<sup>1</sup> tec:inno GmbH

Sauerwiesen 2, D-67661 Kaiserslautern, Germany

[schumacher@tecinnno.com](mailto:schumacher@tecinnno.com)

<sup>2</sup> Center for Learning Systems and Applications (LSA), University of Kaiserslautern

Postfach 3049, D-67653 Kaiserslautern, Germany

[bergmann@informatik.uni-kl.de](mailto:bergmann@informatik.uni-kl.de)

**Abstract.** This paper presents an approach to realize a case retrieval engine on top of a relational database. In a nutshell the core idea is to approximate a similarity-based retrieval with SQL-queries. The approach avoids duplicating the case data or compiling index structures and is therefore ideal for huge case bases which are often subject to changes. The presented approach is fully implemented as part of the commercial CBR toolbox **orange** and the experimental evaluation demonstrates impressive retrieval behaviour.

## 1 Introduction

Case-Based Reasoning (CBR) has become a very important technique for realizing intelligent product recommendation agents for Electronic-Commerce applications [3,8,9]. The core of such applications is a product database that describes the specific features of each available product. When applying CBR, this product database is treated as a case base, i.e., each product record in the database is interpreted as a case in the case base. During the retrieval phase of the CBR cycle, product cases are retrieved based on the similarity to the query that describes the product requirements that are important for the customer. The retrieved products can then be presented to the customer or they may be further customized, e.g., by applying adaptation techniques from CBR [6].

One core problem when developing such E-Commerce applications is to enable the case-based recommendation agent to get *efficient*, *up-to-date*, and *consistent* access to the product database. The typical application scenario is as follows:

- The products are stored in an existing (relational) database that is independent from the recommendation agent. This database is maintained by the supplier of the products and is also used for many other services related to the products.

- The number of products in the database is very huge.
- There are rapid and continuous changes in the product database, in particular when products are individual items (e.g., used cars) that become unavailable when they are sold.
- The representation of products is quite simple, typically a list of attribute values.

To link the product database with the case-based retrieval component, different architectural variants are possible [4].

**Retrieval Inside the Database.** From a purely technical point of view, the ideal approach would be to integrate the similarity-based retrieval function into the database itself. This would enable efficient data access and would ensure an up-to-date and consistent view on the data. In such an ideal scenario, the database would be able to automatically process a similarity-based query formulated in a standardized query format enabling flexible representation of similarity measures. In the recent research literature on databases some very interesting results can be found on indexing structures for similarity-based retrieval [2,1,7]. Unfortunately, these techniques have not yet been introduced into commercially available standard databases. There also exists no standardized language for a similarity-based query formulation that is widely accepted by the database community. Due to the large number of different databases from different vendors that could possibly hold product data, it is currently not feasible to realize the case-based retrieval inside the database in the near future.

**Retrieval on Top of the Database.** Since from a pragmatic point of view the integration of the retrieval function into the database is not feasible at the moment, the second architectural variant must be applied as a medium-term solution. This variant means to realize the similarity-based retrieval on top of the database. The retrieval engine becomes a separate module that interfaces with the database to get access to the product data. Here again, different variants are possible.

- *Bulk-loading all products into the retrieval engine:* This approach replicates the product database inside the retrieval engine. The shortcomings of this approach are that the required storage space is doubled and that it is necessary to update the duplicated case base every time when the product database is updated. Hence, major consistency problems arise.
- *Constructing and storing an index structure for similarity-based retrieval in the retrieval engine:* Here, the problem of duplicating the product data is avoided, but the consistency problem remains.
- *Approximating similarity-based retrieval with SQL-queries:* This third option is a MAC/FAC-like approach. Like proposed in [5], for every query a sequence of SQL queries is constructed to fetch cases from the database which are then used for a detailed similarity assessment. This approach avoids the

consistency problem and always relies on the up-to-date data. The challenge is to develop an approach that maintains efficiency.

In the remainder of this paper, we analyze in detail the problem of approximating similarity-based retrieval by SQL-queries and we presents a new retrieval algorithm which generalizes the results presented by [5]. The presented approach is implemented in Java as a retrieval service for tec:inno's *orange*-framework and will be in use at the time of the workshop. We will give some experimental results of performance measurements in section 4.

## 2 Foundations

We assume that the cases and queries are represented as lists of attribute-value pairs ( $A_1 = a_1, \dots, A_n = a_n$ ). The allowed range or *type*  $T_i$  of attribute  $A_i$  is defined either by a subset of (natural or real) numbers or by a finite set of symbolic values. The value of an attribute in a case must be one element of the type or the special value *undefined*:  $a_i \in T_i \cup \{\text{undefined}\}$ .

Such cases can be easily represented by one table in a relational database: Each attribute corresponds to one column of the table, one row of the table completely describes a single case. An attribute is undefined if the corresponding value in the database table is *null*. The relation between the case structure and the database tables is called a *mapping*. In the following we will identify the attributes with the table columns and denote the columns also by  $A_i$ .

For such cases it is relatively easy to compute a similarity between query and case: For each attribute we can define a local similarity measure which describes the similarity between different attribute values. For numerical attributes this is a monotonous function on the distance of the values,<sup>1</sup> for a symbolic attribute it is defined by a table containing the similarity values. We assume further that the local similarity is 1 if the query attribute is undefined, which expresses a "don't care"-semantic, and 0 if only the case attribute is undefined, which expresses a penalty for the case which lacks a feature the user asked for. The global similarity between two objects can be calculated by accumulating the local similarities of each attribute. In the following we assume that a weighted average of the local similarities is computed, but the considerations also apply to other similar approaches.

Besides a query object, a query can contain additional information for the retrieval: first it can contain *user weights* for each attribute to express which attributes are more or less important for the customer, second the customer can specify *filters* which define hard constraints for the attribute values of the case. Finally, it contains the number of cases to retrieve, let this number be  $k$ .

First we consider a query without weights and filters, also we assume all attribute types to be numeric or at least totally ordered. Then we can view our case base as an  $n$ -dimensional euclidean space in which each object is represented by a point, if all attributes are defined. The  $k$  most similar cases then lie within

<sup>1</sup> The greater the distance, the smaller is the similarity of two values

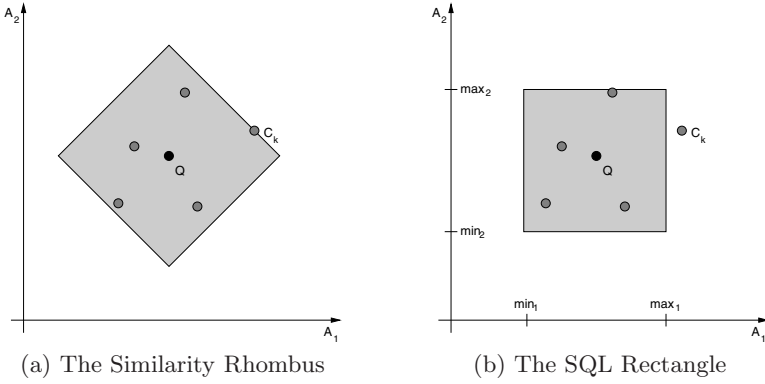


**Fig. 1.** Volumes covered by queries

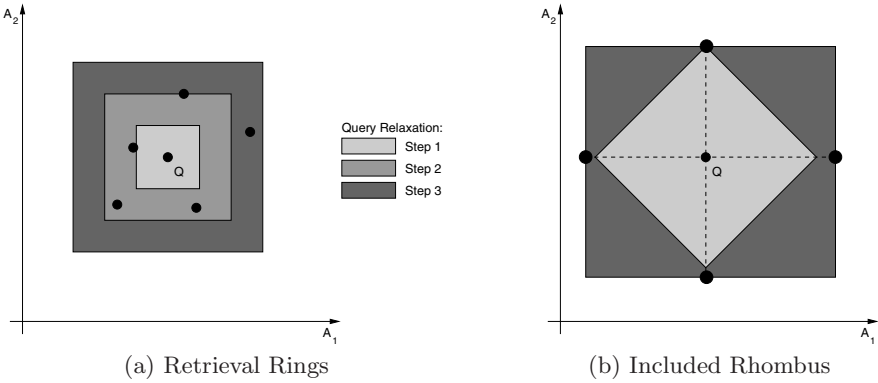


**Fig. 2.** Retrieval with Rings of SQL-Rectangles

a kind of “hyper-rhombus” centered on the query point  $Q$ , where the size of the rhombus is determined by the least similar of the retrieved cases (see figure 1(a)). If we view in the same way a SQL-query of the form

$$\begin{aligned}
 &\text{SELECT } a_1, \dots, a_n \text{ FROM CaseTable} \\
 &\text{WHERE } (a_{i_1} \geq \min_{i_1} \text{ AND } a_{i_1} \leq \max_{i_1}) \dots \\
 &\quad \text{AND } (a_{i_m} \geq \min_{i_m} \text{ AND } a_{i_m} \leq \max_{i_m})
 \end{aligned} \tag{1}$$

where  $a_{i_1}, \dots, a_{i_m}$  are those attributes in the query which are not undefined, we get a “hyper-rectangle” (see figure 1(b)). The goal is now to use the rectangular SQL-queries to retrieve the cases inside the similarity-rhombus from the database.

Our solution is to construct a series of rectangular “rings” around the query point as shown in figure 2(a). In step 0 the retrieval starts with the query point

```

ALGORITHM DBRetrieve
INPUT : Object query, Integer k
OUTPUT : List of Object result[1...k]

BEGIN
  step = 0
  WHILE (NotFinished(query, result)) DO
    sqlQuery = RelaxQuery(query, step, k)
    dbResult = ExecuteSQLQuery(sqlQuery)
    newCases = SortCases(query, dbResult)
    merge(result, newCases)
    step = step + 1
  END

```

**Fig. 3.** Basic Retrieval Algorithm

itself and the cases from the next ring are retrieved as long as more cases are needed to determine the retrieval result.<sup>2</sup> This process is called *query relaxation*. The cases read from the database are ordered according to their similarity to the query, and finally the  $k$  most similar cases are returned. So our technique can be described by the algorithm shown in figure 3.

The function `NotFinished(query, result)` has to implement a suitable criterion to decide if the retrieval can be finished. One criterion could be that the retrieval should always yield a complete result list, i.e., there are no other cases in the case base which are more similar to the query than those in the result list. The similarity rhombus already covered by the SQL-queries is defined by the most similar object lying exactly on the borders of the rectangle (see figure 2(b)). If the least similar case in the result list is more similar to the query than this object then we know that we will not find more similar cases by relaxing the query any further. However, in practice this criterion can easily mean that a huge part of the case base has to be searched, because the object in the corners of the SQL rectangle are less similar to the query the more attributes the query has. Also, for many applications completeness of the search is not that important. Therefore, an alternative criterion is to stop the retrieval if  $k$  results have been found. The tests have shown, that this is sufficient to yield a good result if we make sure that the SQL-queries cover significantly more than  $k$  cases.

The speed of the relaxation is crucial to the performance of the retrieval: If we relax the query too fast, we risk to retrieve too many cases from the database which have to be imported in the search engine and evaluated. This would cost both much main memory and the ordering of the cases would take long. If the relaxation is too slow, we would have to do a lot of SQL-queries. While this may be optimal for memory use and evaluation, it would be slowed

<sup>2</sup> Note that the algorithm can also be used as an “any-time”-algorithm, where the user can examine an intermediate result and trigger another step, until the result is sufficient.

down by the overhead involved with each database query. Often the database runs on another machine than the search engine, and so this would create a lot of network traffic. Therefore we want to retrieve the necessary cases with only a few database queries.

For the following considerations we view the space of cases as a  $n$ -dimensional hyper-cube of volume 1, i.e., all edges of the cube have the length 1. The distance between two values  $x$  and  $y$  of the same type is defined as  $d(x, y) = 1 - \text{sim}_i(x, y)$  where  $\text{sim}_i$  is the local similarity measure for this type. Further we assume that the cases are distributed equally in this space. The goal of the following consideration is to find parameters for the first step of the query relaxation so that in this step already enough cases are retrieved from the database to build an acceptable query result. For this we introduce the parameter  $N_{max}$ , which is the maximum number of cases we want to read from the database in a single step. It clearly has to be a function of  $k$  because we want the  $k$  most similar cases to be included in this number of cases, therefore we write it as a function  $N_{max}(k)$ .

To generate the bounds of the database query we now have to determine which *volume* the query should cover in order to have a result of  $N_{max}(k)$  cases. For a SQL-query this volume can be computed as:

$$V = \prod_{j=1}^m (1 - \text{sim}_{i_j}(\min_{i_j}, \max_{i_j})) \quad (2)$$

If the query object is  $Q = (q_1, \dots, q_n)$  and we select the bounds of the SQL-query so that the speed of the relaxation is equal for all attributes, we get:

$$\begin{aligned} \forall_{j=1}^m : \text{sim}_g &= \text{sim}_{i_j}(q_{i_j}, \min_{i_j}) = \text{sim}_{i_j}(q_{i_j}, \max_{i_j}) \\ \Rightarrow V &= (2 \cdot (1 - \text{sim}_g))^m \end{aligned} \quad (3)$$

We call  $\text{sim}_g$  the *minimal global similarity* because all cases covered by the rectangle are at least this similar to the query. Because we have assumed the cases to be distributed equally we can describe the volume of one case as  $V_C = 1/N$  where  $N$  is the number of cases in the case base, which leads to:

$$V = V_C \cdot N_{max}(k) \Rightarrow \text{sim}_g = 1 - \frac{1}{2} \left( \frac{N_{max}(k)}{N} \right)^{\frac{1}{m}} \quad (4)$$

In this way the minimum global similarity for the first relaxation step  $\text{sim}_g^1 = \text{sim}_g$  depends only on the number of defined attributes in the query, the requested size of the result set and the number of cases in the database.

To construct the SQL-query, the local similarity measures have to be *invertible*, i.e., it must be possible to determine the following set:

$$\text{SIM}_j(q_j, \text{sim}_g) = \{x_j \in T_j \mid \text{sim}_j(q_j, x_j) \geq \text{sim}_g\} \quad (5)$$

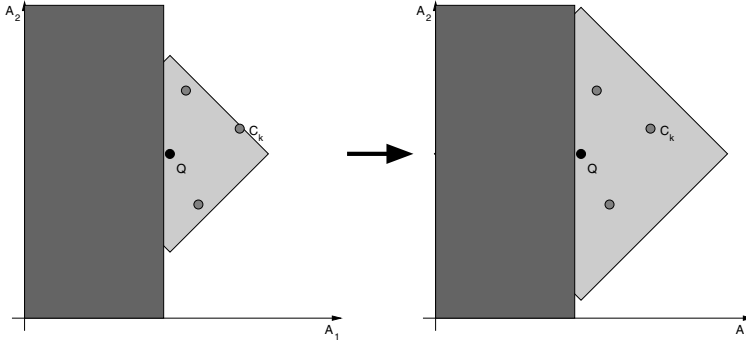


Fig. 4. Effect of a Filter

This set contains all attribute values for which the local similarity to the query is greater than the given minimum global similarity. If the attribute type is totally ordered and the similarity measure is monotonous with respect to the distance of the values, we can also express this set as an interval  $SIM_j(q_j, sim_g) = [min_j, max_j]$ . Both forms of the set can be used easily to construct a SQL-query; the WHERE-Clause of the query in step  $s$  we denote with  $C_s$ .

For further steps  $s$  we can compute  $sim_g^s$  the same way by using a linear growth of the query volume, but this way  $sim_g^s$  would change very slowly for higher  $s$ , which could mean that no new cases are covered by the query. To prevent this and to ensure that eventually all cases of the case base can be reached, we set  $sim_g^s = 1 - s \cdot (1 - sim_g)$ .

Finally, we have to modify the SQL queries so that in each step not the complete rectangle but only the current ring is read from the database. This can be done by excluding the rectangle of the step before in the query:

SELECT attributes FROM table WHERE  $C_s$  AND NOT ( $C_{s-1}$ )

### 3 Advanced Features

Earlier we stated that the similarity of a case to the query is computed as the weighted average of the local similarities of the attributes. Additionally a query can define *user weights* that modify the predefined weights of the attributes, and *filters* that exclude cases with certain attribute values. This all significantly changes the form of the similarity rhombus and therefore has to be taken into account when constructing the database queries.

A filter just cuts off a part of the solution space like shown in figure 4. While in general the effect on the rhombus can be quite complex, we have found that in practical applications often a very special filter type is used: the “more”- or “less”-filter. It specifies that for all resultant cases a certain attribute should

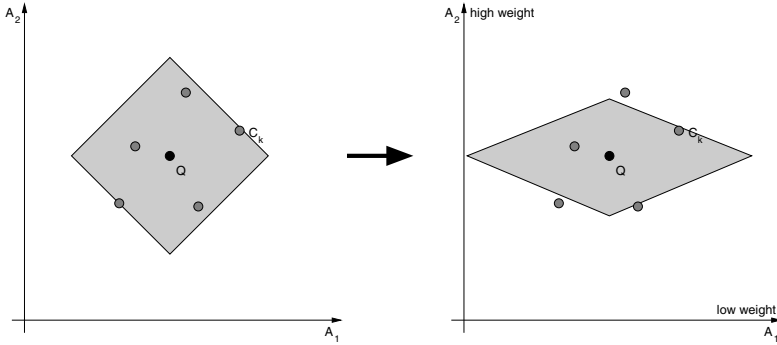


**Fig. 5.** Effect of Weights

have a value greater or smaller than the one defined in the query. This kind of filter is easy to consider: It cuts off exactly one half of the similarity rhombus. We can easily exclude the filtered cases in the SQL-query so that we do not have to read cases which cannot be valid solutions. But this will have the consequence that we get only about half of the number of cases we wanted to be read. To compensate for this, we have to double the query volume for every attribute which is constrained by a filter.

To consider weights, we denote the complete weight of attribute  $A_j$  with  $w_j$ , where the default weight is 1. Different weights for different attributes change the form of the similarity rhombus like shown in figure 5: An attribute with a higher weight has to be matched more closely than one with a lower weight. Therefore, we have to change the form of the SQL rectangle in a similar way. The goal is that the similarity of a case in the corner of the rectangle is the minimum global similarity  $sim_g$ . This can be done by computing for each attribute  $A_j$  an own *minimum local similarity*  $sim_{l,j}$ , which leads to the following equation:

$$sim_{l,j} = 1 + \frac{\sum_j w_j \cdot (sim_l - 1)}{w_j \cdot m} \quad (6)$$

A last problem concerns *underspecified queries*. These are queries which have many exact matches in the database. With the naive approach “load all cases from the database selected by the query” this leads to huge amounts of data to be read from the database. To avoid this, we implemented the import from the database in a way that only  $N_{max}(k)$  cases are imported from the database result set. Should it be necessary to import more cases later, the same result set can be used to read the remaining cases of this query without having to perform a new database query. Of course in general this makes the result almost certainly incomplete, but as it occurs in most cases only with exact matches, this is not a serious problem.



## 4 Evaluation

In this section we want to present some results of experiments which we conducted to examine the performance of our retrieval approach.

The software used for the tests implements all features described in section 3. The first version stops after at least  $k$  cases have been loaded from the database, so it is possible that the retrieval results of this engine are incomplete. A second version of the retrieval engine therefore does further relaxation steps until the completeness of the retrieval result can be ensured. In the following, we will call this version the “complete DB-retrieval”. We compared these two engines to the “linear retrieval” which holds all cases in the main memory without a connection to the database and performs the retrieval by computing the similarity of all cases and selecting the  $k$  best.

For the function  $N_{max}(k)$  used during the query relaxation in the DB retrieval, we chose the following definition:

$$N_{max} = \max \left( N_{min}, N_{min} + N_{incr} \cdot \log_2 \frac{k}{N_{R,min}} \right) \quad (7)$$

It is motivated by the idea that for small  $k$  we want to read relatively more cases from the database in order to ensure that the best  $k$  cases are actually found, while for bigger  $k$  the completeness of the result is not as important as the performance. We got good results with the settings  $N_{R,min} = 10$ ,  $N_{min} = 100$ , and  $N_{incr} = 50$ .

The case base used for the tests contains 8622 cases having 3 symbolic and 4 numeric attributes each. The retrieval engines are implemented in Java and compiled to native code using the IBM HPJ compiler. They work as TCP/IP-servers to which queries can be sent over the network. The queries as well as the results are written in XML. All timings include the sending of queries and receiving of results. As the data base management system we used a Microsoft SQL Server 7.0.

In the first experiment we compared the retrieval times of the different engines for queries with different numbers of defined attribute values. In all tests  $k = 10$  cases were retrieved. We started with two attributes and extended the query in each step by another attribute. Table 1 shows the results of this experiment. Besides the times needed for the retrieval the table also gives the number of SQL-queries executed by the DB-retrieval engines. Finally we determined the “ $\alpha$ -error” of the first version of the DB-retrieval, which is the number of most similar cases *not* found. This gives a measure of the incompleteness of the DB-retrieval.

We see that the linear retrieval takes up to 40 times as long as the DB-retrieval in this experiment. Maybe more surprising is that in all tests the first version of the DB-retrieval-engine produced a complete result, i.e., the result list contained the 10 most similar cases or at least cases which are equally similar to the ones returned by the linear retrieval. Further, we see that in the tests

**Table 1.** Retrieval times with respect to number of attributes defined in the query

| No. Attr. | Linear Retr. (ms) | DB-Retr. (ms)    | Steps | $\alpha$ -Error | Complete DB-Retr. (ms) | Steps |
|-----------|-------------------|------------------|-------|-----------------|------------------------|-------|
| <b>2</b>  | 6701              | $175.9 \pm 49.1$ | 2     | 0               | $150.0 \pm 33.8$       | 2     |
| <b>3</b>  | 7059              | $187.3 \pm 46.1$ | 2     | 0               | $190.0 \pm 51.5$       | 2     |
| <b>4</b>  | 7405              | $205.9 \pm 39.3$ | 2     | 0               | $191.0 \pm 36.5$       | 2     |
| <b>5</b>  | 7656              | $233.9 \pm 45.1$ | 2     | 0               | $401.0 \pm 27.4$       | 3     |
| <b>6</b>  | 8067              | $288.3 \pm 49.2$ | 3     | 0               | $450.0 \pm 44.0$       | 4     |
| <b>7</b>  | 8290              | $306.1 \pm 42.0$ | 2     | 0               | $301.0 \pm 62.4$       | 3     |

with 5, 6, and 7 defined attributes the complete DB-retrieval needed to do one more step to ensure the completeness of the result though no further results were found in this extra step.

From this experiment we find that our retrieval technique has a very good performance. Further we can conclude that, at least for small  $k$ , the possible incompleteness of the DB-retrieval-approach is not critical: Usually the most similar cases are found.

We wanted to verify this conclusion with a further test in which we used the same query but changed the requested result set size  $k$ . The query in this experiment has 6 defined attribute values, and  $k$  is increased from 10 in three steps to 30, 50, and 70. The results are shown in table 2.

**Table 2.** Retrieval times with respect to the result set size

| $k$       | Linear Retr. (ms) | DB-Retr. (ms)    | Steps | $\alpha$ -Error | Complete DB-Retr. (ms) | Steps |
|-----------|-------------------|------------------|-------|-----------------|------------------------|-------|
| <b>10</b> | 8106              | $168.9 \pm 41.1$ | 2     | 0               | $150.0 \pm 46.2$       | 2     |
| <b>30</b> | 8155              | $261.0 \pm 71.1$ | 2     | 0               | $290.0 \pm 44.4$       | 3     |
| <b>50</b> | 8458              | $281.4 \pm 51.5$ | 3     | 4               | $380.0 \pm 33.9$       | 4     |
| <b>70</b> | 8719              | $368.9 \pm 55.5$ | 4     | 22              | $20740 \pm 808$        | 5     |

We see that for small result set sizes like 10 or 30 we get again complete retrieval results from the first version of the DB-retrieval engine. Also in both cases the performance is similar to the results of the first experiment. But for  $k = 50$  we see that there is a incompleteness in the DB-retriever's result: 4 cases of the 50 most similar ones are not found. The complete DB-retriever avoids this by doing one more retrieval step. In the final test with  $k = 70$  the incompleteness is even worse, 22 of the most similar cases are not found. Again the complete retriever tries to prevent this with an additional relaxation step, but in this

cases this leads to loading all cases into the main memory, and therefore the performance of the complete version of the DB-retrieval is very bad.

As a conclusion from this experiment we see that with increasing result set sizes the DB-retrieval can easily produce incomplete results. We also see that while it is possible to prevent this by doing additional relaxation steps this can have the effect of a very bad performance, so usually one will be content with the incomplete result which is retrieved in very short time, because in many domains the performance will be more critical than the completeness. In favour of the “incomplete” version of the DB-retrieval we can give the fact, that in the tests that produced incomplete results, the similarity of the found cases was only slightly less than that of the cases in the complete result. So while the found cases were not the perfect ones, the still were very good in terms of similarity.

Also, the DB-retrieval is quite efficient in terms of memory consumption: While the linear retriever used about 75 MB of memory, the DB-retrieval needed less than 30 MB.

## 5 Discussion and Future Work

Finally we want to give a short discussions of the benefits and shortcomings of our approach. We recognize the following advantages:

- *Efficient Retrieval*: The evaluation indicates that a good retrieval performance can be expected even on very huge case bases.
- *No Consistency Problem*: Each query is answered by using all cases in the database without having to rebuild any index structures or loading cases into the retrieval engine. Cases are available immediately after they are stored in the database. This makes this technique optimal for applications with case bases that change very often.
- *Independence of the Database System*: Any standard SQL database implementation can be used for this technique. No database-specific programming has to be done inside or outside the database.
- *Use of Existing Databases*: If the tables are in the required form they can be used directly for retrieval. Otherwise one can apply a database view to provide the data in a suitable form, but usually nothing has to be changed in the database schema.

On the other hand, there are some limitations that have to be worked on. First, the retrieval technique presented so far is limited to attribute-value representations. Extensions towards representing structured cases are necessary. As a first step in this direction a feature is under development and evaluation using  $n$ -to- $m$ -relations in the database for representing attributes holding sets of values.

Second, the current model is also based on the idealistic assumption of a uniform distribution of the cases on the representation space. If this assumption is violated the efficiency of the retrieval will decrease because either too many cases will be retrieved from the database or too many SQL queries are created.

We expect that this problem can be avoided by incorporating an introspective learning approach that estimates the density of the case base in different regions.

Finally, more evaluation needs to be done in order to learn about the behaviour of the retrieval for different sizes of case bases or different parameters for the functions used in the retrieval. By the time of the workshop we also will be able to report lessons learned from a commercial application in which this retrieval engine is used.

## Acknowledgements

The authors wish to thank the staff of tec:inno for valuable comments during the development of the database retrieval engine.

Funding for this research has been partially provided by the Commission of the European Communities (ESPRIT contract EP 27.068, the WEBSELL project - Intelligent Sales Assistants for the World Wide Web). The partners of WEBSELL are tec:inno (prime contractor, Germany), IMS (Ireland), IWT Magazin Verlags GmbH (Germany), Adwired AG (Switzerland), Trinity College Dublin (Ireland) and the University of Kaiserslautern (Germany).

## References

1. S. Berchtold, B. Ertl, D. A. Keim, H.-P. Kriegel, and T. Seidl. Fast nearest neighbor search in high-dimensional space. In *Proc. 14th Int. Conf. on Data Engineering (ICDE'98)*, pages 209–218, 1998. 274
2. S. Berchtold, D. A. Keim, and H.-P. Kriegel. The x-tree: An index structure for high-dimensional data. In *Proc. 22nd Int. Conf. on Very Large Data Bases*, pages 28–39, 1996. 274
3. R. Bergmann, S. Breen, M. Göker, M. Manago, and S. Wess. *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology*. Lecture Notes in Artificial Intelligence, State-of-the-Art-Survey, LNAI 1612. Springer Verlag, 1999. 273
4. M. Donner and T. Roth-Berghofer. Architectures for integration CBR-systems with databases for e-commerce. In *7th German Workshop on Case Based Reasoning (GWCBR'99)*, 1999. 274
5. H. Shimazu, H. Kitano, and A. Shibata. Retrieving cases from relational databases: Another strike toward corporate-wide case-based systems. In *Proceedings of the 13th International Joint Conference in Artificial Intelligence (IJCAI'93)*, 1993. 274, 275
6. S. Schmitt and R. Bergmann. Applying case-based reasoning technology for product selection and customization in electronic commerce environments. In *12th Bled Electronic Commerce Conference.*, 1999. 273
7. T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 154–165, 1998. 274
8. W. Wilke. *Knowledge Management for Intelligent Sales Support in Electronic Commerce*. DISKI 213. Infix Verlag, 1999. 273

9. W. Wilke, M. Lenz, and S. Wess. Intelligent sales support with CBR. In M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess, editors, *Case-Based Reasoning Technology: From Foundations to Applications*, pages 91–113. Springer Verlag, 1998. 273

# Maintaining Case-Based Reasoning Systems Using Fuzzy Decision Trees<sup>1</sup>

Simon Chi Keung Shiu, Cai Hung Sun, Xi Zhao Wang and Daniel So Yeung

Department of Computing, Hong Kong Polytechnic University

Hung Hom, Kowloon, Hong Kong

{csckshiu, cschsun, csxzwang, csdaniel}@comp.polyu.edu.hk

**Abstract.** This paper proposes a methodology of maintaining Case Based Reasoning (CBR) systems by using fuzzy decision tree induction - a machine learning technique. The methodology is mainly based on the idea that a large case library can be transformed to a small case library together with a group of adaptation rules, which are generated by fuzzy decision trees. Firstly, an approach to learning feature weights automatically is used to evaluate the importance of different features in a given case-base. Secondly, clustering of cases will be carried out to identify different concepts in the case-base using the acquired feature knowledge. Thirdly, adaptation rules will be mined for each concept using fuzzy decision trees. Finally, a selection strategy based on the concepts of  $\varepsilon$ -coverage and  $\varepsilon$ -reachability is used to select representative cases. The effectiveness of the method is demonstrated experimentally using two sets of testing data.

## 1 Introduction

Case-base maintenance is the process of refining a CBR system's case-base to improve the system's performance, i.e., case-base maintenance implements policies for revising the organization or contents (representation, domain content, accounting information, or implementation) of the case-base in order to facilitate future reasoning for a particular set of performance objectives [9]. In the past, researchers have attempted to address various aspects of the case-base maintenance problem. To provide maintenance support at the case level, Smyth and Keane [12] suggested a competence preserving deletion approach. Competence (or coverage) is the range of target problems that a given system can solve, and is also a fundamental evaluation criterion of CBR system performance. Barry Smyth et al. [13] also presented a new model of case competence, and demonstrated a way in which the proposed model of competence can be used to assist case authors. Anand et al. [1] proposed to use data mining techniques in a novel role of a back-end technology for case-based reasoning (CBR) systems, i.e., the acquisition of cases and discovery of adaptation knowledge.

---

<sup>1</sup> This project is supported by a HK PolyU grant PA25

Hanney and Keane [5] presented an inductive learning algorithm to extract adaptation knowledge from the cases in the case-base. Their algorithm builds pairs of cases and uses the feature differences of these case pairs to build adaptation rules which are very useful in case-base maintenance. The approach of Hanney et al. is based on the assumption that the differences occurred between cases in the case-base are representative of the differences that will occur between future problems and the case-base.

Although many researchers, as aforementioned, have contributed to solving the case-base maintenance problems, there is still no systematic method available. In this paper, we propose a systematic methodology for case-base maintenance, which integrates identifying salient features, distinguishing different concepts, learning adaptation knowledge, computing case competence and selecting seed cases together into a framework of case-base maintenance. The methodology is mainly based on a process that a large case library is transformed to a small case library together with a group of adaptation rules, which are generated by fuzzy decision trees. The details of maintaining a case-base from scratch, as proposed in this paper, consists of four steps. Firstly, an approach to learning feature weight automatically is used to evaluate the importance of different features in a given case-base. Secondly, clustering of cases will be carried out to identify different concepts in the case-base using the acquired feature knowledge. Thirdly, adaptation rules will be mined for each concept using fuzzy decision trees. Finally, a selection strategy based on the concepts of  $\varepsilon$ -coverage and  $\varepsilon$ -reachability is used to select representative cases. The effectiveness of the method is demonstrated experimentally on the Rice Taste Data and the Boston Housing Data. The result shows that the testing case-base size can be reduced by 39% and 43% respectively if we complement the remaining cases by adaptation rules discovered using our approach. The overall competence of the smaller case-bases is 90% of the original ones.

## 2 Methodology of Maintaining a Case Library

Throughout this section, we consider a case library in which all features are supposed to take numeric real values. It should be noted that the real-valued features discussed here could be, without difficulties, extended to the features, which take values in a normed vector space. We first introduce a weighted distance metric and a similarity measure used in the following.

Let  $CL = \{e_1, e_2, \dots, e_N\}$  denote a case library. Each case in the library can be identified by an index of corresponding features. In addition each case has an associated action. More formally we use a collection of features  $\{F_j \ (j = 1, \dots, n)\}$  to represent the cases and a variable  $V$  to denote the action. The  $i$ -th case  $e_i$  in the library can be represented as a  $n+1$ -dimensional vector, i.e.  $e_i = (x_{i1}, x_{i2}, \dots, x_{in}, v_i)$  where  $x_{ij}$  corresponds to the value of feature  $F_j$  ( $1 \leq j \leq n$ ) and  $v_i$  corresponds to the action ( $i = 1, \dots, N$ ).

Suppose that for each  $j$  ( $1 \leq j \leq n$ ) a weight  $w_j$  ( $w_j \in [0,1]$ ) has been assigned to the  $j$ -th feature to indicate the importance of the feature. Then, for any pair of cases  $e_p$  and  $e_q$  in the library, a weighted distance metric can be defined as

$$d_{pq}^{(w)} = d^{(w)}(e_p, e_q) = \left( \sum_{j=1}^n w_j^2 (x_{pj} - x_{qj})^2 \right)^{1/2} = \left( \sum_{j=1}^n w_j^2 \chi_j^2 \right)^{1/2} \quad (1)$$

where  $\chi_j^2 = (x_{pj} - x_{qj})^2$ . When all the weights are equal to 1 the distance metric defined above coincides with the Euclidean measure, denoted by  $d_{pq}^{(1)}$ , in short, denoted by  $d_{pq}$ .

By using the weighted distance defined in equation (1), a similarity measure between two cases,  $SM_{pq}^{(w)}$ , can be defined as follows:

$$SM_{pq}^{(w)} = \frac{1}{1 + \alpha \cdot d_{pq}^{(w)}} \quad (2)$$

where  $\alpha$  is a positive parameter. When all weights take value 1 the similarity measure is denoted by  $SM_{pq}^{(1)}$ .

After introducing the weighted distance metric and the similarity measure, our methodology, which consists of four major phases, will be described in the following four sub-sections.

## 2.1 Phase One - Learning Feature Weights

In this section, a feature evaluation function is defined. The smaller is the evaluation value, the better are the corresponding features. Thus we would like to find the weights such that the evaluation function attains its minimum. The task of minimization of the evaluation function with respect to weights is performed using a gradient decent technique. We formulate this optimization problem as follows.

For a given collection of feature weights  $w_j$  ( $w_j \in [0,1]$ ,  $j = 1, \dots, n$ ) and a pair of cases  $e_p$  and  $e_q$ , equation (1) defines a weighted distance measure  $d_{pq}^{(w)}$  and equation (2) defines a similarity measure  $SM_{pq}^{(w)}$ . When all weights take value 1,  $d_{pq}^{(w)}$  and  $SM_{pq}^{(w)}$  will degenerate the Euclidean distance  $d_{pq}^{(1)}$  and  $SM_{pq}^{(1)}$ . A feature evaluation index  $E$  is defined as



$$E(w) = \frac{2 * [\sum_p \sum_{q(q < p)} \left( SM_{pq}^{(w)} (1 - SM_{pq}^{(1)}) + SM_{pq}^{(1)} (1 - SM_{pq}^{(w)}) \right)]}{N * (N - 1)} \quad (3)$$

where  $N$  is the number of cases in the case base.

Noting that the feature evaluation function  $E(w)$  will gradually become zero when  $SM_{pq}^{(w)} \rightarrow 0$  or  $1$ , we hope finding a collection of weights such that the feature evaluation function attains its minimum.

To minimize equation (3), we use a gradient decent technique. The change in  $w_j$  (i.e.  $\Delta w_j$ ) is computed as

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j}, \quad (4)$$

for  $j = 1, \dots, n$ , where  $\eta$  is the learning rate.

For the computation of  $\frac{\partial E}{\partial w_j}$ , the following expressions are used:

$$\frac{\partial E(w)}{\partial w_j} = \frac{2 * [\sum_p \sum_{q(q < p)} \left( 1 - 2 \cdot SM_{pq}^{(1)} \right) \cdot \frac{\partial SM_{pq}^{(w)}}{\partial d_{pq}^{(w)}} \cdot \frac{\partial d_{pq}^{(w)}}{\partial w_j} ]}{N * (N - 1)} \quad (5)$$

$$\frac{\partial SM_{pq}^{(w)}}{\partial d_{pq}^{(w)}} = \frac{-\alpha}{(1 + \alpha \cdot d_{pq}^{(w)})^2} \quad (6)$$

$$\frac{\partial d_{pq}^{(w)}}{\partial w_j} = \quad (7)$$

The training algorithm is described as follows.

Step 1. Select the parameter  $\alpha$  and the learning rate  $\eta$ .

Step 2. Initialize  $w_j$  with random values in  $[0, 1]$ .

Step 3. Compute  $\Delta w_j$  for each  $j$  using equation (4).

Step 4. Update  $w_j$  with  $w_j + \Delta w_j$  for each  $j$ .

Step 5. Repeat step 3 and step 4 until convergence, i.e., until the value of  $E$  becomes less than or equal to a given threshold or until the number of iterations exceeds a certain predefined number.

After training, the function  $E(w)$  attains a local minimum. We expect that, in average, the similarity values  $\{SM_{pq}^{(w)}, p = 1, \dots, N, q < p\}$  with trained weights are closer to 0 or 1 than that without trained weights such as  $\{SM_{pq}^{(1)}, p = 1, \dots, N, q < p\}$ .

## 2.2 Phase Two - Partitioning the Case Library into Several Clusters

This section attempts to partition the case library into several clusters by using the weighted distance metric with the weights learned in section 2.1. Since the considered features are considered to be real-valued, many methods such as K-Means clustering [2] and Kohonen' self-organizing network [8] can be used to partition the case library. However, this paper adopts a typical approach of clustering, i.e., similarity matrix [4] which uses only the information of similarity between cases. This approach first transform the similarity matrix to a equivalent matrix and then considers the cases being equivalent each other as one cluster. The procedure is as follows.

- Step 1. Give a significant level (threshold)  $\beta \in (0,1]$
- Step 2. Determine the similarity matrix  $SM = \left( SM_{pq}^{(w)} \right)$  according to equation (2) and (1).
- Step 3. Compute  $SM1 = SM \circ SM = \left( s_{pq} \right)$  where
 
$$s_{pq} = \max_k \left( \min(SM_{pk}^{(w)}, SM_{kq}^{(w)}) \right).$$
- Step 4. If  $SM1 \subset SM$  then go to step 5, else replace SM with SM1 and go to step 3.
- Step 5. Determine several clusters based on the rule "case p and case q belong to the same cluster if and only if  $s_{pq} \geq \beta$ ".

## 2.3 Phase Three - Mining Adaptation Rules by Fuzzy Decision Trees

In this paper we follow the hypothesis that each cluster contain cases that are more representative of others and a non-representative case in the cluster is considered as a perturbation of certain representative case. The perturbation is handled by a group of adaptation rules. In details, let p be a representative case and q be a non-representative case in some cluster, we expect that the solution of q can be obtained by an appropriate adaptation (adjustment) of the solution of p according to some adaptation rules. For example, a representative case and a non-representative case are respectively  $p = (1, 2, 3, 4)$  and  $q = (0.9, 2, 3, 4.01)$  in which the first three

components are feature-values and the last component is the solution, an adaptation rule is “IF the difference between the feature values is negatively small THEN the adjustment of the solution is very positively small”, then q’s solution can be considered to be obtained by p’s solution with an adjustment based on the adaptation rule.

Fuzzy decision tree generation technique such as ID3, which is based on minimum information entropy to select expanded attributes, was proposed by Quinlan in 1986 [11]. Subsequently, the fuzzy version of ID3 based on minimum fuzzy entropy was suggested by several authors [6,7,14]. The second approach has many advantages over the first such as robustness and comprehensibility. That motivates the choice of this technique.

For each cluster  $L = \{e_1, e_2, \dots, e_m\}$  obtained from phase 2, we denote its cases in the form of  $e_i = (x_{i1}, x_{i2}, \dots, x_{in}, v_i)$  where  $x_{ij}$  corresponds to the value of feature  $F_j$  ( $1 \leq j \leq n$ ) and  $v_i$  corresponds to the action ( $i = 1, \dots, m$ ). Arbitrarily taking a case  $e_k$  ( $1 \leq k \leq m$ ) in the cluster L, a set of vectors namely  $\{f_i \mid f_i \in R^{n+1}, i = 1, 2, \dots, m\}$  can be computed in the following way

$$f_i = e_i - e_k = (x_{i1} - x_{k1}, x_{i2} - x_{k2}, \dots, x_{in} - x_{kn}, v_i - v_k) = \{y_{i1}, y_{i2}, \dots, y_{in}, u_i\}$$

We attempt to find several adaptation rules with respect to the case  $e_k$  ( $1 \leq k \leq m$ ) from the set of vectors  $\{f_i \mid f_i \in R^{n+1}, i = 1, 2, \dots, m\}$  by fuzzy decision tree.

Consider a problem of learning from examples in which there are  $n+1$  numerical attributes,  $\{A^{(1)}, A^{(2)}, \dots, A^{(n)}, A^{(n+1)}\}$  ( $A^{(n+1)}$  is the classification attribute). Then  $\{f_i \mid i = 1, 2, \dots, m\}$  can be regarded as  $m$  examples described by the  $n+1$  attributes. We first fuzzify these  $n+1$  numerical attributes into linguistic terms.

The number of linguistic terms for each attribute is assumed to be five (which can be enlarged or reduced if it is needed in a real problem). These five linguistic terms are Negative Big, Negative Small, Zero, Positive Small, and Positive Big, in short, NB, NS, ZE, PS and PB respectively. Their membership functions are supposed to have triangular form and shown in Figure 1. For each attribute (the  $j$ -th attribute  $A^{(k)}$ ,  $1 \leq k \leq n+1$ ) with the attribute-values  $\text{Range}(A^{(k)}) = \{y_{1k}, y_{2k}, \dots, y_{mk}\}$ , the two parameters in Figure 1, a and b, are defined by

$$a = \sum_{y \in N} y / \text{Card}(N) \quad \text{and} \quad b = \sum_{y \in P} y / \text{Card}(P) \quad (8)$$

in which  $N = \{y \mid y \in \text{Range}(A^{(k)}), y < 0\}$ ,  $P = \text{Range}(A^{(k)}) - N$  and  $\text{Card}(E)$  denotes the cardinality of a crisp set E.

After this fuzzification, each example can be regarded as a  $5 \times (n + 1)$ -dimensional vector of membership degree. By putting the  $m$  vectors together, a matrix with  $m$  rows and  $5 \times (n + 1)$  columns is formed. According to this matrix, we propose our fuzzy decision tree generation procedure – fuzzy ID3. In comparison with the existing versions of fuzzy ID3, our proposal is founded on such a viewpoint that each linguistic term of attributes and each node in the tree are considered to be fuzzy sets defined on the example-label space  $\{1, 2, \dots, m\}$ .

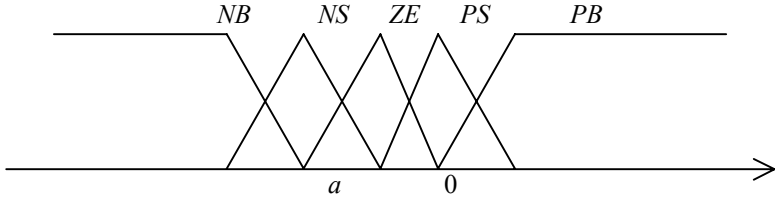


Fig. 1. Five membership functions

After generating the fuzzy decision tree by using fuzzy ID3 algorithm, a set of adaptation rules can be extracted from the tree. The extraction is straightforward, i.e., each path from the root to a leaf is converted into an adaptation rule (fuzzy production rule). Thus the number of leaves is just the number of adaptation rules.

## 2.4 Phase Four - Selecting Representative Cases

This phase aims to select representative cases from each cluster according to the adaptation rules obtained in phase three. Our selection strategy is based on a  $\varepsilon$ -coverage concept. Instead of the deletion, this paper proposes a selection strategy which makes use of Smyth proposed concepts of coverage and reachability with some changes (called  $\varepsilon$ -coverage and  $\varepsilon$ -reachability respectively).

Let  $L$  be a cluster in which each case  $e$  is accompanied with a set of adaptation rules  $AR(e)$ ,  $\varepsilon$  be a small positive number,  $e_p = (x_{p1}, x_{p2}, \dots, x_{pn}, v_p)$  and  $e_q = (x_{q1}, x_{q2}, \dots, x_{qn}, v_q)$  be two cases in the cluster  $L$ . According to the reasoning mechanism established in phase 3, an adjustment amount  $\Delta$  of the solution for the case  $e_q$  can be obtained by matching  $(x_{q1} - x_{p1}, \dots, x_{qn} - x_{pn})$  against  $AR(e_p)$ . If  $v_q + \Delta \in (v_p - \varepsilon, v_p + \varepsilon)$ , then  $e_p$  is said to  $\varepsilon$ -cover with  $e_q$ . The  $\varepsilon$ -coverage and  $\varepsilon$ -reachability of the case  $e_p$  are defined by

$$\text{Coverage}(e_p) = \{e \mid e \in L, e \text{ is } \varepsilon\text{-covered by } e_p\} \quad (9)$$

and

$$\text{Reachability}(e_p) = \{e \mid e \in L, e \text{ } \varepsilon\text{-covers with } e_p\} \text{ respectively} \quad (10)$$

respectively.

The  $\varepsilon$ -coverage of a case  $e$  represents the generalization capability of this case. The bigger is the number of cases in the  $\varepsilon$ -coverage, the more representative is the selected case  $e$ . As a kind of rule extraction, it is a commonsense application of Occam's Razor. On the other hand, the  $\varepsilon$ -reachability of a case  $e$  represents the degree to which  $e$  can be replaced by another case. The smaller is the number of cases in the  $\varepsilon$ -reachability, the more important is the selected case  $e$ . As an index of evaluation of selected cases, to a great extent, it reflects the difference between rule-based and case-based approaches. Our proposed selection procedure integrates these two approaches together.

3 Experimental Analysis

This section presents the experimental analysis of our methodology on two real-world problems that are taken from [10] and [3], i.e. the rice taste (RT) and the Boston housing (BH) problem. The RT data consists of five inputs and a single output whose values are associated with subjective evaluations of the flavor, appearance, taste, stickiness, toughness and overall evaluation of 105 different kinds of rice (table one show some typical records). The BH data consists of thirteen inputs and a single output whose values are taken from 506 housing records in suburbs of Boston. The experiments were carried out in a Pentium III 500MHz machine and the programs were all written in Microsoft Visual C++.

3.1 The Rice Taste Data

Table 1. Rice Taste Datat sizes of headings

| Favor  | Appearance | Taste  | Stickiness | Toughness | Overall Evaluation |
|--------|------------|--------|------------|-----------|--------------------|
| 0.699  | 1.543      | 1.76   | 1.944      | -0.875    | 1.706              |
| -0.593 | -0.898     | -0.883 | -0.647     | 0.323     | -1.235             |
| 0.158  | 0.163      | 0.03   | 0.359      | -0.128    | 0.135              |

After applying the learning feature weight algorithm mentioned in section 2.1 to this cases, the feature weight results shown in Table 2 were obtained (Learning iterations = 10,000 cycles, Alpha = 0.46 and Learning rate = 0.05).

Table 2. Feature weights

| Favor | Appearance | Taste | Stickiness | Toughness | Overall Evaluation |
|-------|------------|-------|------------|-----------|--------------------|
| 0.02  | 0.03       | 0.54  | 0.03       | 0.04      | 2.68               |

The clustering of cases is carried out both with and without the feature weight information. The results are shown in Table 3. When feature weight information is used to guide the clustering of cases, much better results were obtained, i.e. the number of clusters are between 9 to 26 and there are at least 70% of the cases which fall in the first 5 largest clusters. The user can specify a particular significant level ( $\beta$ ) for further learning of adaptation rules.

**Table 3.** Clustering of cases with and without feature weights

| Significant level ( $\beta$ ) | Without feature weights |                                          |                                           | With feature weights |                                          |                                           |
|-------------------------------|-------------------------|------------------------------------------|-------------------------------------------|----------------------|------------------------------------------|-------------------------------------------|
|                               | No. of clusters         | No. of clusters contains one case record | Cases in the first 5 largest clusters (%) | No. of clusters      | No. of clusters contains one case record | cases in the first 5 largest clusters (%) |
| 0.86                          | 99                      | 94                                       | 10.48                                     | 9                    | 4                                        | 96.19                                     |
| 0.87                          | 100                     | 96                                       | 9.52                                      | 14                   | 5                                        | 84.76                                     |
| 0.88                          | 102                     | 100                                      | 7.62                                      | 15                   | 6                                        | 83.81                                     |
| 0.89                          | 102                     | 100                                      | 7.62                                      | 20                   | 9                                        | 79.05                                     |
| 0.90                          | 104                     | 103                                      | 5.71                                      | 26                   | 13                                       | 70.48                                     |

In our experiment, we have chosen Beta = 0.87 as the significant level. As a result, the cases are partitioned into 14 clusters.

In the mining of fuzzy adaptation rules, the first five problem features, i.e. Favor, Appearance, Taste, Stickiness and Toughness, were fuzzified into three linguistic variables, Small, Middle, Big. The solution feature, i.e. Overall Evaluation, was fuzzified into five linguistic variables, i.e. Negative Big, Negative Small, Zero, Positive Small, Positive Big.

The general form of a fuzzy adaptation rule generated from the fuzzy decision tree is as follow:

IF the difference of X1 is [Small | Middle | Big]

[AND the difference of X2 is [Small | Middle | Big]

[AND the difference of X3 is [Small | Middle | Big]]]

THEN the difference of overall evaluation is [Negative Big | Negative Small | Zero Positive Small | Positive Big ].

Where  $X = \{\text{Favor, Appearance, Taste, Stickiness, Toughness}\}$  and the maximum number of antecedents of each fuzzy rule is limited to three in this experiment. For example, in cluster 5, which consists of 13 cases, one of the adaptation rules is:

**Table 4.** Case coverage

| Case number (x) | Number of cases which are covered by case(x) | The actual cases which are covered by case(x) | No. of adaptation rules |
|-----------------|----------------------------------------------|-----------------------------------------------|-------------------------|
| 1               | 4                                            | 3,4,9,11                                      | 6                       |
| 2               | 6                                            | 3,6,9,10,11,12                                | 11                      |
| 3               | 2                                            | 7,9                                           | 11                      |
| 4               | 4                                            | 6,9,11,12                                     | 10                      |
| 5               | 5                                            | 1,2,6,9,11                                    | 10                      |
| 6               | 3                                            | 1,7,11                                        | 12                      |
| 7               | 8                                            | 1,3,5,6,9,10,11,12                            | 16                      |
| 8               | 3                                            | 2,9,11                                        | 14                      |
| 9               | 1                                            | 8                                             | 15                      |
| 10              | 6                                            | 4,6,7,9,11,12                                 | 10                      |
| 11              | 4                                            | 3,6,9,12                                      | 14                      |
| 12              | 3                                            | 4,6,11                                        | 13                      |
| 13              | 2                                            | 1,11                                          | 11                      |

Rule1: IF the difference of Favor is small  
AND the difference of Toughness is small  
AND the difference of Taste is small  
THEN the difference of overall evaluation is Zero.  
The rule's confidence is 0.92

According to the selecting case policy defined in section 2.4, we select cases {7,2,4,8,13} as the representative cases in this cluster 5 (see Table 4). The overall selection results of cluster 3, 4, and 5 are shown in Table 5 where for cluster 3,  $\epsilon=0.15$  (i.e. the absolute error of expected value is not greater than 0.15), for cluster 4,  $\epsilon=0.1$  and for cluster 5,  $\epsilon=0.15$ .

**Table 5.** Overall selection result of cluster 3, 4 and 5

| Cluster No | Number of cases | No. of Representative cases | No. of deleted cases |
|------------|-----------------|-----------------------------|----------------------|
| 3          | 34              | 18                          | 16                   |
| 4          | 30              | 13                          | 17                   |
| 5          | 13              | 5                           | 8                    |
| Total      | 77              | 36                          | 41                   |

Therefore, the total number of deleted cases is 41. After removing these deleted cases, we use them back to test the competence of the smaller case-base with the adaptation rules. The average relative error of the solution in each cluster is shown as in Table 6. We defined relative error = (real value – expected value)/real value \*100%.

The result shows that the testing case-base size can be reduced by 39% if we complement the remaining cases by adaptation rules discovered using our approach. The overall competence of the smaller case-base is 90% of the original one.

**Table 6.** Average error after deletion

| Cluster No    | The average relative error of the deleted cases in this cluster |
|---------------|-----------------------------------------------------------------|
| 3             | 8.76%                                                           |
| 4             | 14.86%                                                          |
| 5             | 3.48%                                                           |
| Average error | 10.26%                                                          |

### 3.2 The Boston Housing Data

Using the similar approach as described in section 3.1, the Boston housing data are successfully partitioned into 93 clusters, with 13 clusters having at least 9 cases and the biggest cluster consists of 47 cases. However there are also 56 very small clusters with only one case each. Learning of fuzzy adaptation rules are carried out to those 37 clusters, (i.e. clusters having more than one case), and a selection policy is adopted with  $\epsilon=0.10$ . The result shows that the case-base size can be reduced by 43% and the overall competence of the smaller case-base is 90% of the original one.

## 4 Summary and Future Works

In this paper, we have developed a methodology of maintaining Case Based Reasoning systems by using fuzzy decision tree induction. The main idea is to transform a large case library to a small case library together with a group of adaptation rules, which are generated by fuzzy decision trees. These adaptation rules play the role of complementing the reduction of cases. The experimental analysis of our method showed promising results. Future work include (1) a large scale testing of our methodology using different case-bases, (2) different selection polices will be developed based on ideas such as subsumption, conflicting and ambiguity which exist among cases and (3) A reasoning method will be developed for effective problem solving using this hybrid knowledge representation scheme. This involves the development of similarity computation of new cases versus the fuzzy rules/small case-library knowledge structure in the case base. This reasoning method should also have the ability to adapt the retrieved solution to suit the new case.

## References

1. Anand, S. S., Patterson, D., Hughes, J. G. and Bell D. A., "Discovering Case Knowledge using Data Mining," in Second Pacific Asia Conference, PAKDD-98, Australia, pp. 25-35.
2. Bezdek, J. C. , "Pattern recognition with fuzzy objective function algorithms," Plenum, NewYork, 1981.
3. Boston Housing Data, UCI ML Repository, [www.ics.uci.edu/~mllearn/MLRepository.html](http://www.ics.uci.edu/~mllearn/MLRepository.html)
4. Fu, G., "An algorithm for computing the transitive closure of a similarity matrix," Fuzzy Sets and Systems, vol. 51, pp. 189-194, 1992.
5. Hanney, K. and Keane, M. T., "Learning Adaptation Rules from a Case-Base," In Proc. Advances in Case-Based Reasoning, 3rd European Workshop, EWCBR-96, pp179-192.
6. Ichihashi, H. et al., "Neuro-fuzzy ID3, " in Fuzzy Sets and Systems vol. 81, pp.157-167, 1996.
7. Jeng, B. et al., "FILM: a fuzzy inductive learning method for automated knowledge acquisition," in Decision Support Systems, vol. 21, pp. 61-73, 1997.
8. Kohonen, T., "Self-Organization and Associate Memory", Springer, Berlin, 1988.
9. Leake, D. B. and Wilson, D. C., "Categorizing Case-Base Maintenance: Dimensions and Directions," Advances in Case-Based Reasoning, 4th European Workshop, EWCBR-98 , pp. 196-207.
10. Nozaki, K., Ishibuchi, H. and Tanaka, H, "A simple but powerful heuristic method for generating fuzzy rules from numerical data," in Fuzzy Sets and Systems, FSS 86(1997), pp. 251-270.
11. Quinlan, J. R., "Induction of decision trees", Machine Learning, vol. 1, pp.81-106, 1986.



12. Smyth, B. and Keane, M. T. , “Remembering to Forget: A Competence-Preserving Case Deletion Policy for Case-based Reasoning systems”, Proceedings of the fourteenth International Joint Conference on Artificial Intelligence, IJCAI-95, pp. 377-382.
13. Smyth, B. and Mckenna, E., “Modeling the Competence of Case-bases”, Advances in Case-Based Reasoning, 4th European Workshop, EWCBR-98, pp. 23-25.
14. Umanol, M. et al., “Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems,” in IEEE International Conference on Fuzzy Systems, (26-29 June 1994), pp. 2113-2118.

# Applying Recursive CBR for the Customization of Structured Products in an Electronic Shop

Armin Stahl and Ralph Bergmann

Centre for Learning Systems and Applications (LSA), University of Kaiserslautern  
PO-Box 3049, D-67653 Kaiserslautern, Germany  
{stahl,bergmann}@informatik.uni-kl.de

**Abstract.** When applying CBR for Electronic Commerce, the adaptation capabilities of CBR can be used for product customization. Most adaptation techniques suffer from the problem that they require a large knowledge acquisition effort which leads to problems in the rapidly changing E-Commerce scenario. In this paper we present a new approach to adaptation that is particularly suited to Electronic Commerce applications. It assumes that products can be structured hierarchically into sub-components. Adaptation is achieved by incrementally replacing unsuitable sub-components through recursively applying CBR to find best-matching alternative sub-components. The presented approach avoids huge portions of the knowledge acquisition effort and is prototypically implemented as an extension of the CBR-Works tool.

## 1 Introduction

With the increasing number of companies and private persons who get fast access to the World Wide Web (WWW), Electronic Commerce is playing more and more a central role in economy. Case-Based Reasoning (CBR) has become a very important technique for realizing intelligent product recommendation agents for such E-Commerce applications [1,10,4]. While for fixed, unchangeable products a pure similarity-based retrieval approach is sufficient, product *customization* is important when complex products with a large number of possible variants must be supported during sales [8]. Examples of such products are technical products (e.g., personal computers) or service products (e.g., insurances or investment plans). Product customization can be realized by the known adaptation techniques from CBR. However, the applicability of the different techniques strongly depends on the kind of products to be supported, in particular on the number of different variants. A general problem in particular for the transformation-based approaches to adaptation is that they require a large effort for acquiring the adaptation knowledge, to be represented, for example, as rules or as adaptation operators. This turns into a real problem when products have many different variants or if the product spectrum changes rapidly so that the adaptation knowledge must be updated. On the other hand, derivational approaches to adaptation require a complete problem solver for product recommendation and

sufficient knowledge for this problem solver. This approach also suffers from intractability, both in terms of computational efficiency and knowledge acquisition effort. Hence, for customizing complex products with a large number of possible variants, the existing adaptation approaches from CBR are often not suitable in practice.

In this paper we present a new approach to adaptation that is particularly suited to E-Commerce applications. This approach avoids huge portions of the knowledge acquisition effort of the previous approaches. It assumes products that are structured into sub-components, possibly in a hierarchical manner. The knowledge required is knowledge about available pre-configured complete products as well as knowledge about available sub-components. Both kinds of knowledge are easily available in an E-Commerce setting. After retrieving the best pre-configured product with respect to the customers requirements, the product is customized by incrementally replacing sub-components by more suitable sub-components. These new sub-components are determined by recursively applying CBR, i.e. similarity-based retrieval, on the level of the sub-components.

In the remainder of this paper, this approach is described in more detail. We will first describe a typical E-Commerce scenario and analyze the shortcomings of existing approaches. Then the general idea of recursive CBR is introduced before it is specialized for the purpose of product customization. Finally, we report on the current state of the implementation.

## 2 Product Configuration in E-Commerce Applications

### 2.1 Case-Based Product Recommendation: The Scenario

We consider the scenario that a customer enters a virtual shop to buy a complex technical system. Complex technical systems (e.g., PCs) can usually be decomposed into a set of different components. We suppose that the shop offers a set of pre-configured standard systems as well as the different individual components.

When providing an intelligent product recommendation agent, the first step of the sales process is a demand acquisition phase in which the customer states his individual demands on the searched product. The result of this demand acquisition phase can be formulated in form of an incomplete product description (on a technical level) which we call *query*. This query is then used to start a similarity-based retrieval for determining one of the available pre-configured standard systems that fulfills the demands as well as possible. However, because of the large number of possible product variants, the retrieved product usually does not fulfill the demands exactly. Hence, to be able to present the customer a satisfying result, it is necessary to customize the product, e.g., by replacing some components by more suitable ones. This is the task of the adaptation phase of CBR. In the following we review several well known approaches to adaptation in CBR and discuss whether they are appropriate for the customization of complex products.

## 2.2 Existing Approaches

**Transformational Adaptation.** One approach for realizing case adaptation is known under the term *transformational adaptation* [11]. Here, the knowledge how existing solutions can be adapted on actual problems is encoded explicitly, e.g., in form of rules. *Adaptation rules* [3] consist of a set of preconditions and a set of actions. Dependent on the evaluation of the preconditions the actions are able to modify a retrieved case to a new target case with respect to the given query. *Adaptation operators* [8] are very similar to adaptation rules. While the rules are executed after the retrieval automatically, the operators provide more control by the customer. Therefore, the retrieved case with an additional set of applicable operators is presented to the customer. If the retrieved product does not fulfill his demands, he can repeatedly choose operators to change the given product until he gets a satisfying result.

The general problem of both approaches is the necessity to define every possible case modification that may occur explicitly in the form of preconditions and actions. Therefore, even simple domains often require a large number of adaptation rules or operators to cover all adaptation possibilities. For really complex domains the huge amount of necessary rules or operators prevents the application in practice.

**Configuration from Scratch.** The configuration of a product can also be performed from-scratch by classic configuration systems [5] without using CBR. Applying such a system for the customization of products in an E-Commerce application often leads to some disadvantages. First, if it is impossible to configure a product that fulfills all customer demands exactly, such a system rarely finds a suitable alternative solution. Second, in classic configuration systems it is often difficult to handle optimality criteria. The system can only present any solution (if existing), but a high quality of this solution cannot be guaranteed in general. An additional problem of the from-scratch configuration is the time-critical aspect, i.e., the configuration process usually takes a lot of time. This is often not acceptable in E-Commerce applications.

**Derivational Adaptation.** The common property of adaptation rules and operators is that they are used to transform an existing solution to a new, hopefully better solution. Another adaptation approach is known under the term *derivational adaptation*. Here, the cases do not represent concrete solutions. Instead, they contain a description of the problem solving trace which describes how a concrete solution is generated. This known solution trace can then be used by a from-scratch problem solver to generate a solution very quickly. Generally, this approach is coupled with the same disadvantages as configuration from-scratch, except for better performance. A description of a configuration system using derivational adaptation can be found in [7].

In the next section we will introduce the basic theoretical idea for developing a new approach to realize efficient product configuration by using CBR.

### 3 Adaptation of Highly Structured Cases

#### 3.1 Highly Structured Problems

Consider the situation that we have a domain in which it is possible to subdivide the arising problems and the related solutions into more or less independent sub-problems and related sub-solutions. This means, the complete problems, or we can also call them *complex problems*, can only be solved if all included sub-problems are solved and combined to an overall solution. When assuming such a problem structure, it is possible to represent the problems and the corresponding solutions in form of part-of hierarchies (see figure 1). The root-nodes of such part-of hierarchies represent the complex problems or solutions, respectively. Inner nodes represent *complex sub-problems* (*sub-solutions*) that can be divided again into less complex sub-problems (sub-solutions). Finally, the leaf nodes represent *atomic problems* and the related *atomic solutions*.

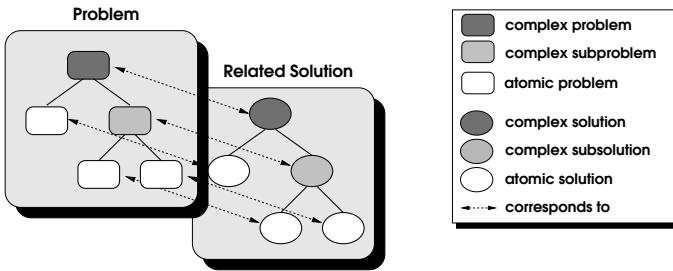


Fig. 1. Highly Structured Problems

**Dependencies between Sub-Problems.** Generally, the single sub-problems cannot be considered absolutely isolated because of dependencies between them. The solution of one sub-problem often depends on the solutions of some other sub-problems. Thus, it is unavoidable to handle these dependencies while solving the different sub-problems, or at least when combining them to the final solution.

#### 3.2 The Idea of Recursive Case-Based Reasoning

Assuming a highly structured domain as described in the previous section, we can now introduce a special procedure for solving such problems by applying CBR. The first step is equal to the retrieval phase of the classic CBR-cycle. The given (complex) problem is used as query and a most similar case is retrieved from a case-base. Due to the complexity of the problem structure, in general, it is unlikely that the complete solution of the retrieved case can be applied to the current problem directly. This means that the retrieved case can include sub-solutions which cannot be reused. Then it is necessary to adapt these unsuitable sub-solutions. Such adaptation can become a very complex task if the unsuitable sub-solutions belong to complex sub-problems.

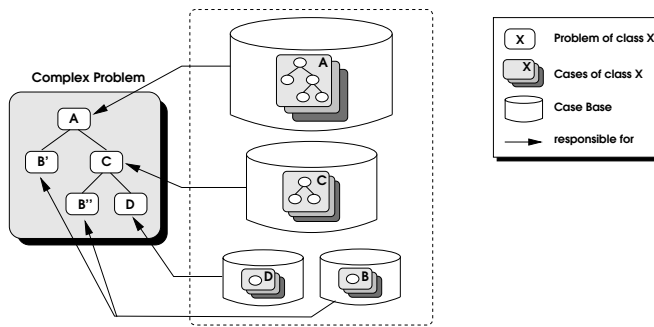


**Fig. 2.** The Use of Several Case-Bases

The idea of recursive CBR is to realize the necessary adaptation again in a case-based manner. This means that we will not try to adapt the retrieved but unsuitable sub-solutions by modifying them with traditional methods. Instead, we use still unsolved sub-problems as new queries for a retrieval in additional case-bases. These case-bases must contain cases that correspond to the classes of the current sub-problems, i.e., the cases are generally less complex than the cases in the top-level case-base (see figure 2). We now distinguish two situations, depending on the position of an unsolved sub-problem in the hierarchy:

- If the unsolved sub-problem is an atomic problem, the responsible case-base contains atomic solutions only. In this case we suppose that retrieving a case that can be reused directly is possible. Nevertheless, if adaptation is still necessary, it should be quite simple so that it can, for example, be realized by applying few adaptation rules.
- When having an unsolved complex sub-problem and retrieving a suitable case is impossible, we can repeat the described procedure in a recursive manner. This means that we start a retrieval to find a suitable sub-solution. If the retrieved complex sub-solution cannot be reused directly, necessary adaptation can be realized again in the described case-based manner.

To obtain a solution for the overall complex problem which was the starting point of the whole process, it is finally necessary to combine all found sub-solutions. An illustration of the suggested procedure is shown in figure 3.

The basic idea of recursive CBR is the principle of “Divide and Conquer”, which can often be applied successfully in Computer Science. Generally, it is important to consider one basic limitation of this approach. If it is possible to divide a complex problem into sub-problems that can be solved separately in an easy way, it must also be guaranteed that the final combination of the found sub-solutions will not become too difficult. Otherwise, such a procedure will not be very efficient. Therefore, we assume a limited number of dependencies between the different sub-problems in the application domain to apply our approach.

Another work that applies incremental case adaptation to compose a solution for an actual design task out of the solutions of several cases is the system

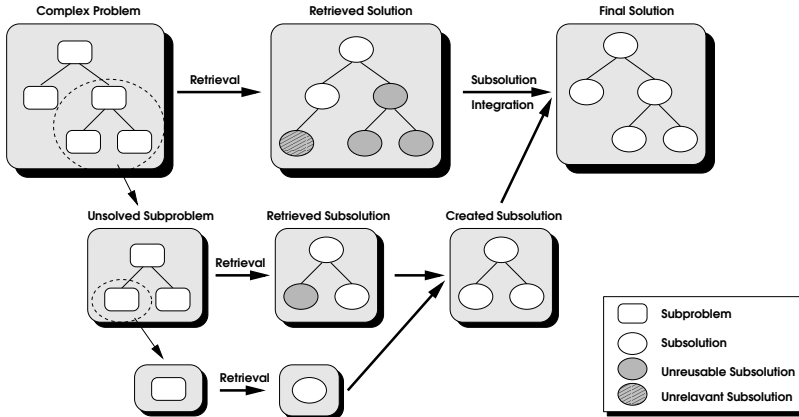


**Fig. 3.** The Idea of Recursive CBR

EADOCS [6]. However, this approach does not follow a problem decompositional hierarchy.

The advantage of a divide and conquer approach for solving problems with large complex cases is described in [9]. It is shown, that dividing a large problem into several smaller sub-problems leads to smaller solution spaces that have to be covered by the different case bases. Therefore, the CBR-System is able to find solutions with a significantly higher accuracy.

## 4 Applying Recursive CBR for Product Configuration

### 4.1 Domain Representation

In the previous section we have presented the idea of recursive CBR. Now, we will show how this idea can be applied for the customization of technical systems. To illustrate the functionality of this approach we consider the configuration of personal computers. However, the presented approach can be applied for the customization of any structured products.

**Structuring Product Cases.** In general, it is possible to represent complex structured products by an object-oriented domain model with an additional aggregation hierarchy [2,3]. In the following we assume a special form of such a compositional structure like shown in figure 4.

In this structure, the root node represents the whole technical system to be configured, i.e., in our example a complete personal computer. The leaf nodes represent the concrete parts that a PC consists of, like the hard-disk, the processor, the CD-Drive, and so on. In a concrete PC these parts are realized by concrete technical components. For example, the processor part can be realized by a Pentium-III with 500MHz. The inner nodes of the compositional structure

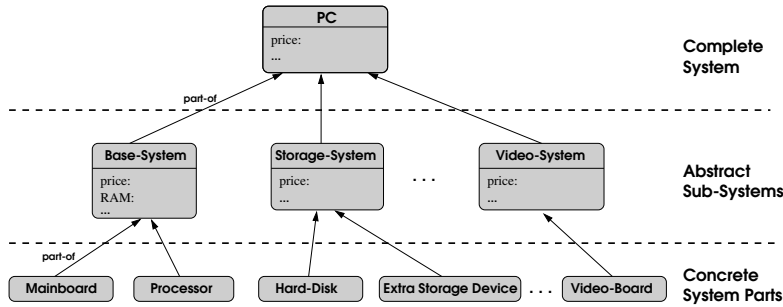


Fig. 4. An Example Compositional Structure

(if existing) do not correspond to concrete parts of the technical system. They rather represent abstract nodes used for aggregating related parts into kinds of abstract subsystems (e.g. Base-System or Storage-System).

**Structured Query.** The starting point of the configuration process is the query represented by an incomplete instantiation of the compositional structure. In section 3.1 we assumed that a complex problem can be decomposed to apply the idea of recursive CBR. When looking at the example query shown in figure 5, we can interpret the root note as our complex problem, i.e., we are searching a PC with a set of special properties. To reach this goal it is necessary to select appropriate components that fulfill the properties of the respective part-queries. In our example, one part-query states that the PC should have a hard-disk with 12GB of capacity. To fulfill this demand we can, e.g., integrate the concrete hard-disk “Maxtor91303D6” for the hard-disk part in the PC. Thus, we can interpret the different leaf nodes of the query as the sub-problems that must be solved to solve the overall problem, i.e., the configuration of the required PC.

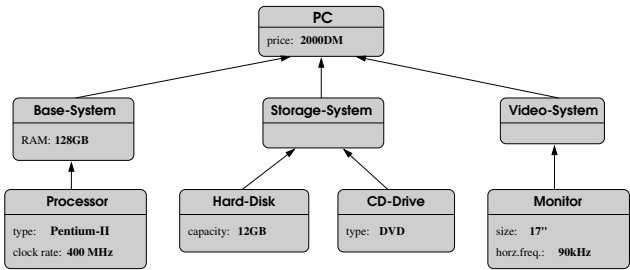


Fig. 5. An Example Query



**Similarity Measure.** To retrieve appropriate PCs from a case-base, the assumed CBR-system uses special domain dependent similarity measures. To compute the similarity between a query and a given product we assume a bottom-up strategy using the defined compositional structure. First, it is necessary to compute the similarities between the part-queries and the corresponding components. These *local similarities* are used for determining the *global similarity* between the query and the product. A more detailed description of such a similarity computation can be found in [2].

**Dependencies between Components.** In section 3.1, we have mentioned the problem of existing dependencies between sub-problems. In the described application domain such dependencies occur in form of technical *constraints* between the different components. For example, it is impossible to combine every kind of processor with any mainboard model because the respective interfaces must fit together. To be able to handle these technical restrictions during the configuration process, the domain model must include a formal description of the existing constraints. For our approach we suppose the existence of a special *constraint system* that is able to check whether the constraints of a given configuration are fulfilled.

4.2 The Configuration Framework

We now describe the different steps during product customization. Figure 6 shows the respective process model.

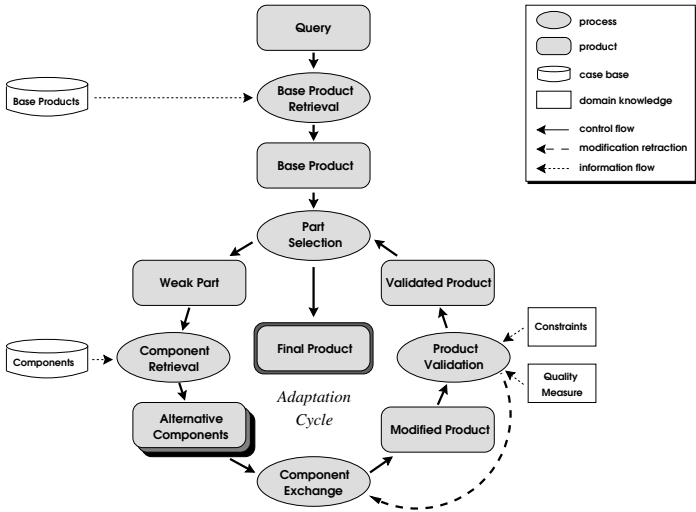


Fig. 6. The Configuration Process

Basically, the whole configuration process can be subdivided into two major steps, called *base product retrieval* and *adaptation cycle*. The task of the first step is the similarity-based retrieval of the best available base product from the respective case-base. The second step is an iterative procedure which performs the necessary adaptation of the retrieved base-product if it does not fulfill all customer demands.

At the beginning of the adaptation cycle, it has to be determined if the base product includes parts that do not fulfill the requirements of the query. This can be done by an examination of the local similarities between the different product parts and the related part-queries. Product parts with a low similarity to the respective part-query are called *weak parts*. The determination of these parts is the task of the *part selection* process. If the product includes no weak parts at all, the adaptation process is finished and the given product is presented to the customer as suggested customization result.

Generally, we can distinguish between two causes for weak parts. The first one occurs if the retrieved standard product does not include components for required product parts. For example, if a customer wants a PC with a modem, but the base-product includes no modem at all. Here, we can speak of a *missing component*. The second cause is given if installed components do not fulfill the technical requirements of the customer, e.g., if the PC still contains a modem but this modem provides only an insufficient data transmission rate. Then, we can speak of a *weak component*.

During the adaptation cycle, a selected weak part leads to a new similarity-based retrieval, called *component retrieval*. In this process, the part-query corresponding to the selected weak part is used to retrieve a collection of alternative components from the component case-base. The elements of this collection are arranged by their similarity to the query in a decreasing order.

The task of the next process is then the replacement of the weak component (if existing) by the first element of the collection of alternative components. If the weak part corresponds to a missing component, the first alternative component is introduced as a new component for this part. Because of the mentioned order, this alternative component represents the most suitable component that is available within the component case-base. The result of the described *component exchange* is a *modified product*.

To judge the effects of the component exchange, the modified product must finally be inspected during the *product validation* process. The task of this process is the validation of the modified product with respect to two different aspects. On the one hand, it must be guaranteed that the modification leads to a consistent product, i.e., to a product fulfilling the technical *constraints* of the respective application domain. Therefore, the assumed constraint system has to check the modified product. On the other hand, the modified product should naturally represent a better solution as the product before the modification. Therefore, we have to compare the similarity to the query of these two products. Either if the modified product is inconsistent or if it has a lower similarity to the query

as the product before the modification, retracting the last modification becomes necessary.

The *retraction* of the last modification is realized by the repeated application of the component exchange process. This means, the just now installed component that had led to the failure of the product validation is replaced by the next element of the alternative components collection. If no available alternative component is suitable to fulfill the mentioned criteria, it becomes necessary to restore the situation at the beginning of the current iteration of the adaptation cycle. In this case, an adaptation of the respective weak product part is currently impossible.

If the product validation could be passed successfully, one iteration of the adaptation cycle is finished. The result of this adaptation cycle is the *validated product*. In the next step, it is again the task of the part selection process to determine a weak part within this validated product which shall be used as the starting point for a new iteration. If it is impossible to determine a weak part whose adaptation could perhaps improve the product, the complete configuration process is succeeded and the final product can be presented to the customer. Optionally, it can be useful, e.g., if the actual configuration result is dissatisfying, to try an additional configuration attempt with another retrieved base product.

### 4.3 Controlling the Adaptation Cycle

After giving an overview of our configuration framework, we will now show that the part selection process (see figure 6) is the most important step of the adaptation cycle. This process is mainly responsible for the overall success of the adaptation procedure.

Generally, we can notice that the adaptation cycle has to solve a combination of a *constraint satisfaction problem (CSP)* and an *optimization problem*. On the one hand, it has to find a combination of different components representing a working product, i.e., a product that fulfills all constraints of the application domain. On the other hand, it has to find an optimal combination fulfilling the demands of the customer as well as possible. To solve this special task it is not sufficient to determine weak product parts during the part selection process in any order. It is rather necessary to find adaptation orders which will lead to an optimal configuration result. Generally, we can distinguish between two different approaches to determine adaptation orders:

**Preservation of Consistency** Up to now we have assumed that a product modification during one iteration of the adaptation cycle must even lead to a consistent product. This means that only adaptation orders that preserve the consistency of the product are allowed.

**Temporary Loss of Consistency** In contrast to the previously described approach it is also possible to allow the temporary loss of consistency during the adaptation of a base product. That means, if the product validation process determines the violation of constraints after the component exchange process, this must not necessarily lead to an immediately retraction procedure.

It is rather possible only to notice the loss of consistency and to continue the whole adaptation process without the retraction of the last modification. But, to find a final solution that can be presented to the customer it is of course necessary to re-establish the consistency during the further adaptation.

The advantage of the first approach is the quite simple implementation. Besides that, it represents an *any time algorithm*, i.e., it allows the interrupt of the adaptation cycle at any time to obtain the best consistent solution that could be formed so far. Nevertheless, with preservation of consistency it is often impossible to get optimal configuration results. The reason are the different constraints between the components. For example, if we want to install a SCSI-hard-disk we have perhaps also to replace the IO-controller. Therefore, the integration of the SCSI-hard-disk will lead to an inconsistent PC before the necessary IO-controller is installed, too. If we allow a temporary loss of consistency during the adaptation we are able to handle such problems. But, this requires more complex algorithms to guarantee the final consistency of the configured product.

## 5 Conclusions and State of Implementation

In our opinion, the most important benefit of the presented approach is the capability to realize case-based configuration without requiring a lot of additional domain knowledge. In contrast to the application of adaptation rules/operators our approach needs only the knowledge about existing components and the constraints between them. It is not necessary to describe every thinkable adaptation-step. The selection of adaptation-steps is controlled by the knowledge that is encoded in already existing similarity measures. The additional case information, i.e., the description of the available components, is not critical because it is usually accessible in existing product databases.

Because the presented approach can obviously be classified as transformational adaptation, it is also coupled with the respective advantages: It is able to configure products if the customer demands are not satisfiable and it can handle implicit optimality criteria through the cases that are assumed to be good pre-configured standard products.

To evaluate the functionality of our configuration approach we have implemented a generic prototype for the described configuration process. This prototype consists of an extension of the commercial CBR tool CBR-Works which has been developed jointly by the University of Kaiserslautern and tec:inno GmbH. To be accessible over the World Wide Web the prototype also provides two respective interfaces for the demand acquisition and the result presentation<sup>1</sup>.

In the current version we have modeled the already mentioned PC-domain as a first test domain. However, in the future it could be useful to apply the system on other application domains to get more concrete experiences about the power of the approach. Because of the structure of the selected PC-domain, currently

---

<sup>1</sup> <http://minsk.informatik.uni-kl.de:8103/launch/WebInterface>

we have only implemented a simple adaptation control algorithm preserving the consistency of the retrieved product after every adaptation step. Therefore, the next step should be the implementation and the evaluation of different more or less sophisticated adaptation control strategies to improve the adaptation process.

## References

1. R. Bergmann, S. Breen, M. Göker, M. Manago, and S. Wess. *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology*. State-of-the-Art-Survey, LNAI 1612. Springer Verlag, 1999. 297
2. R. Bergmann and A. Stahl. Similarity measures for object-oriented case representations. In B. Smyth and P. Cunningham, editors, *Case-Based Reasoning Research and Development*, volume 1488 of *LNAI*, pages 25–36. Springer, 1998. 302, 304
3. R. Bergmann, W. Wilke, I. Vollrath, and S. Wess. Integrating general knowledge with object-oriented case representation and reasoning. In *4th German Workshop: Case-Based Reasoning - System Development and Evaluation 1996*, 1996. 299, 302
4. R. Burke. The wasabi personal shopper: A case-based recommender system. In *11th International Conference on Innovative Applications of Artificial Intelligence*, 1999. 297
5. R. Cunis, A. Günther, and H. Strecker, editors. *Das PLAKON-Buch*. Number 266 in *Informatik-Fachberichte*. Springer, Berlin, 1991. 299
6. B. D. Netten and R. A. Vingerhoeds. Incremental adaptation for conceptual design in EADOCs. In *European Conference on Artificial Intelligence (ECAI'98) Workshop on Adaptation in Case-Based Reasoning*, 1996. 302
7. K. Pfitzner. *Fallbasiertes Konfigurieren technischer Systeme*. Dissertation, Universität Hamburg, 1993. 299
8. S. Schmitt and R. Bergmann. Product customization in an electronic commerce environment using adaptation operators. In *7th German Workshop on Case-Based Reasoning*, 1999. 297, 299
9. I. Watson and S. Perera. Context guided retrieval. In David B. Leake and Enric Plaza, editors, *Case-Based Reasoning Research and Development*, volume 1266 of *Lecture Notes in Artificial Intelligence*, pages 359–370. Springer, July 1997. International Conference on Case-Based Reasoning ICCBR-97. 302
10. W. Wilke. *Knowledge Management for Intelligent Sales Support in Electronic Commerce*. DISKI 213. Infix Verlag, 1999. 297
11. W. Wilke and R. Bergmann. Techniques and knowledge used for adaptation during case-based problem solving. In *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA-98*, 1998. 299

# Handling Vague and Qualitative Criteria in Case-Based Reasoning Applications

Ivo Vollrath

Artificial Intelligence/Knowledge Based Systems group, University of Kaiserslautern  
vollrath@informatik.uni-kl.de  
<http://wwwagr.informatik.uni-kl.de/>

**Abstract.** In some case-based reasoning (CBR) applications for decision support there are a number of vague or qualitative criteria that have to be taken into account by the similarity measure. Sometimes, these criteria are very hard to acquire or quantify and they often conflict with the “main” quality criterion that is measured by the similarity function. Surprisingly, this sometimes is true even for cost criteria (mostly believed to be quite quantitative): in certain applications, the acceptable cost limit depends mostly on the quality that is available and thus cannot be specified a priori. This paper discusses the problems arising from this kind of *implicit* criteria and shows approaches of how they can be integrated into the similarity measure of a case-based reasoning system without the need of artificially quantifying them.

## 1 Introduction

### 1.1 Implicit Cost Criteria within Reuse Applications

Electronic design reuse currently is an area of very high research interest. Specification and retrieval of reusable *intellectual properties* (IP) is a non-trivial task and there have been attempts to develop intelligent IP catalogs providing knowledge-based selection support [1,2,3]. There is a specific problem that arises when such a system is meant to be used commercially (as opposed to using it only within a single company): it is not only the *functionality* and *quality* of the design that is to be measured, but also the *costs* of purchase must be taken into account to make a decision on the IP to choose.

It turns out that a user’s cost criteria are most often not exactly known. The user her/himself would not be able to put them in concrete terms because they depend on the objects available and their qualities. Also, the IP vendor usually does not have clear pricing policies, or is not willing to reveal these policies to the broker (pricing not only depends on the amount of units to be produced, but also on the customer herself and her relation to the vendor). So, even if the user had a definite cost limit, it would be almost impossible to verify that it was obeyed.

In these applications, the user typically is looking for complex components for new cutting-edge development tasks, so she cannot expect to find an IP

that fully satisfies her needs. Fulfillment of the technical requirements has top priority. Costs are only important in that unnecessary costs are to be avoided, but since there are no exact matches regarding functional requirements, it is not a practical solution to first retrieve all matching designs (there are none) and then select the cheapest ones in a second step. So, modeling cost criteria in this context is not a trivial task.

The only vaguely known, and *implicit* cost criteria must be somehow accounted for by the selection support system. This requires suitable representation schemes in the first place. This is a problem that has not been dealt with in great detail by past CBR systems. Although, of course, many systems have to deal with cost criteria, the costs that these systems consider are not as vaguely defined as in the domains this paper is concerned about.

## 1.2 Other Qualitative Criteria

Similar to the cost criteria just mentioned, there are other domains where only vaguely known, and *implicit* criteria must somehow be accounted for. These criteria have the quality of being secondary objectives, only to be taken care of when the other, primary, criteria have been fulfilled as much as possible.

Imagine a customer (young, married, male) looking for a family car. The request for a “family car” can be quantified by specifying a “big trunk” and “little fuel consumption”. The customer will specify additional requirements like the price he is prepared to pay. But also, the customer wants that car to be sporty. “How sporty?” the CBR query interface wants to know. – He cannot say. “Just as sporty as a family car can be.” How can this requirement be specified in a query and accounted for by a similarity measure?

If there were a “sportiness” attribute which could be set to something from, say, “not sporty” to “extremely sporty”, none of these values would express what the customer means, since he does not know in advance what degree of sportiness he can expect. Most CBR systems will probably end up in setting the “sportiness” attribute to “extremely sporty” and significantly reducing the importance of this attribute to avoid it dominating the other criteria that really count. – But that’s not what the customer said: “sportiness” is *quite* important to him, but only “after” the other requirements are fulfilled as much as possible.

This paper discusses different ways to represent and handle knowledge about qualitative secondary criteria. These methods have been developed especially for the cost criteria of electronic reuse applications, but they can be transferred to other decision support applications, like the currently very popular field of electronic commerce (see the family-car example). Section 2 briefly discusses an explicit approach of common practice for modeling this qualitative knowledge, while the methods described in Sect. 3 do not require any artificial quantification to express qualitative requirements. Section 4 compares the strengths of the two different approaches, and sketches open questions and possible extensions.

## 2 Explicitly Representing Costs under Vague Knowledge

A straight-forward way to model cost criteria in a CBR system is to introduce a dedicated (possibly complex) attribute that holds cost-related information about each case in the case base. This approach can be appropriate if a certain minimal amount of pricing information is available. In the simplest case, the cost of the component can be stored in the case as a fixed numerical value, but for reasons explained in Sect. 1.1 this is not possible for the applications we are mostly interested in.

In the case that the vendor has a more complicated pricing policy the costs of a component could still be assessed if the policy was well defined and if the vendor was willing to expose it to the developer of the CBR system. (The latter is very unlikely except if vendor and developer are the same person or company and if the system is considered to be secure in that the pricing policy cannot be deduced by unfriendly users.) If all this holds the cost of a component can be deduced from a number of basic attributes. In particular, the costs depend on the identity of the customer. This turns costs into a *secondary attribute* [4] but whose value depends on the case *and* the query. In [5] this is called a *virtual attribute*.

If there are no fixed prices and the pricing policies are not known, a feasible approach might be to define a limited number of pricing categories. These categories have to be of a qualitative nature and cannot be used to calculate the final costs of a component. However, they *can* be used to estimate the costs in relation to other components. This makes it possible to distinguish expensive cases from cheaper ones. Since in many cases the user does not have detailed pricing requirements, anyway, this approach can be useful.

All these methods of cost representation have in common that they only work well if the user explicitly specifies his/her cost criteria with the query. If s/he would not do this, it would be impossible to decide whether the cost of a given case is acceptable (high similarity) or not: the similarity measure for the “cost” attribute cannot be defined independently of the user’s cost criteria.

**Example Implementation.** The latter described method of explicitly representing costs has been used in the first prototype of the READEE<sup>1</sup> project. This prototype offers functionality-oriented selection support for electronics designers in search for a digital signal processor (DSP) that suits the needs of their application design [6]. During the knowledge acquisition phase for the prototype, it turned out to be impossible to acquire the prices of DSPs for all different kinds of customers. Finally, we resorted to defining only four very rough pricing

---

<sup>1</sup> READEE [‘redl’]: Reuse Assistant for Designs in Electronics Engineering; funded by the foundation ‘innovation’ (Stiftung Innovation) of Rhineland-Palatinate, Germany; Corporate partners: Artificial Intelligence/Knowledge-Based Systems group at the University of Kaiserslautern, engineering office “Dr. Peter Conradi & Partner”, and tec:inno GmbH, Germany.



categories from A (cheapest) to D (most expensive). It turned out that most test users felt comfortable with this kind of classification and the similarity measure was indeed able to separate expensive DSPs from cheaper ones – provided that other quality criteria did not too dominantly vote for expensive processors. However, the focus of interest when developing the prototype was on other issues, so we were satisfied with these results and did not make more detailed tests on the cost sensitivity of the similarity measure. The READEE prototype can be tested at <http://wwwagr.informatik.uni-kl.de/~readee/>.

### 3 Representing Implicit Cost Criteria

The request for high quality on the one hand and low costs on the other hand are two contradicting demands. The previous section showed how these different demands can be separately handled by explicitly representing them as different attributes. The drawback of this approach is that the user is forced to specify a requirement (the pricing limit) that she would rather not specify. If she does not specify a limit, the system must revert to a default limit of the lowest possible price. In effect, the resulting cost similarity is mathematically combined with the other quality-related measures. This is shown schematically in Fig. 1: the query requests some level of quality. The quality-related similarity is 1 if this target quality is met but decreases for cases with a lower quality. The cost-related similarity is 1 only for the cheapest case available (assuming lowest quality is cheapest<sup>2</sup>) and drops continuously to 0 for the most expensive (best) cases.

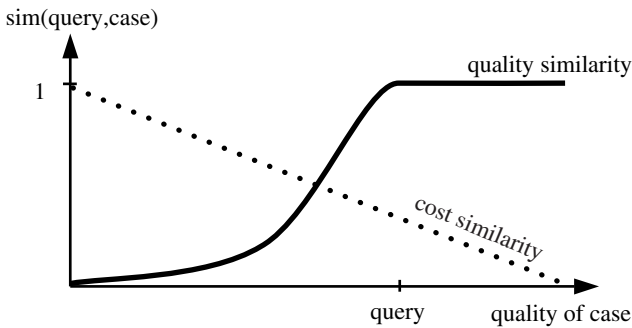


**Fig. 1.** Combining quality similarity with an uninformed cost similarity

For most similarity measures (like weighted sums or products) this imposes two major problems:

<sup>2</sup> This is not always true but a good rule of thumb in most cases. Anyway, Fig. 1 is only meant to illustrate the general idea.

- The maximum combined similarity can never be 1 but depends on the quality of the query. Therefore, similarity values from different queries are not comparable.
- The combined similarity has two maxima because the high cost similarity of some very cheap cases might dominate their low quality similarity.

To overcome these problems it would be beneficial to make the cost similarity not more sensitive than necessary. Indeed, the real cost criteria can be implicitly deduced from the other requirements and can be informally specified as follows:

“Costs may be as high as necessary to fulfill the quality requirements.”

This idea is illustrated in Fig. 2: the cost similarity only drops below 1 if the quality of the case is beyond the requirements of the query. We call this abstract concept of a minimum cost value that is necessary to get the required quality “Point of necessary cost (PNC)”. It should be clear that the PNC is really nothing more than a concept. Only in very rare applications it will be possible to associate this concept with a real monetary value.

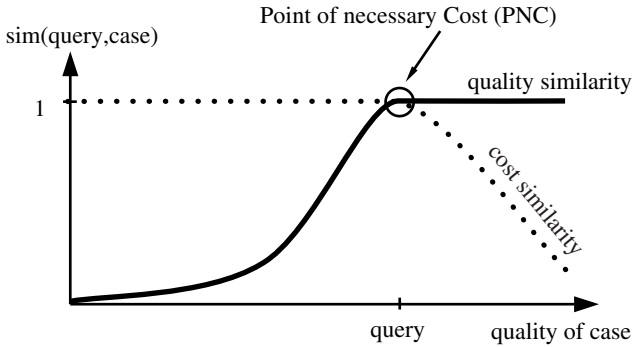


**Fig. 2.** The “ideal” similarity measure: costs may be as high as necessary

The rest of this section shows some ideas about how a good guess for the PNC can be made and how this can be used to reduce the above mentioned problems. Finding a good guess for the PNC eliminates the need to explicitly specify quantified target values for attributes that are more naturally specified in a qualitative way.

Since it is not possible to calculate “the correct” value of the PNC it is useful to think about the implications of over-estimating and under-estimating it. If too high a value is assumed for the PNC there will be a “plateau” of indifferent similarity. This is often undesirable. For many applications it is better to knowingly under-estimate the PNC than to have a similarity measure that is indifferent for the most interesting cases. However, choosing the PNC too low will reduce the benefit of our approach.

### 3.1 Properties of Similarity

It is helpful to analyze requirements that should be met by the similarity measure of a selection support application with implicit cost criteria. It seems reasonable to impose the following requirements on such a similarity measure.

1. Of two cases that fully meet the primary criteria, the cheaper one has a higher similarity.
2. Of two cases that have the same price, the one that better meets the primary criteria has a higher similarity.
3. A case has a maximal similarity if all of the following conditions hold:
  - (a) The case truly fulfills all functionality (primary criteria) specified in the case;
  - (b) the case is not more expensive than “necessary” to fulfill all primary criteria.

These requirements are not formal. Especially, there is some room for interpretation of the word “necessary” in the last condition. Generally, the similarity measure should reward quality only up to the specified requirements and it should give penalties for costs only if they are unnecessarily high.

### 3.2 Concurrent Models of Similarity

The basic idea to approaching a similarity measure that has the properties defined above is to define one similarity measure for each aspect: the *primary* similarity measure expresses the suitability of the attribute with respect to the user’s primary requirements, while the *secondary* similarity measure reflects the acceptability of costs for the current case. These similarity measures must be combined in a suitable way to yield the joined similarity regarding all different aspects of similarity. Because of property 3b the second similarity measure must adapt itself to the primary requirements, i. e. even high costs must not have a negative impact on similarity if the costs are “necessary” to fulfill the primary requirements.

Let us assume that the functional requirements can be measured by individually comparing a number of independent attributes, i. e. the primary similarity measure is a combination of independent similarities on a number of attributes. Let us further assume that the costs, as well, can be expressed as a combination of contributions from a number of independent attributes.<sup>3</sup> The latter set of attributes typically is a subset of the former, since costs are related to functionality. So, each attribute relevant for the primary selection criteria may be also relevant for the secondary cost criteria. The primary and secondary similarity measures now can both be defined as a combination of “local” similarity measures.

---

<sup>3</sup> This is often not true but in Sect. 3.5 we will see how this problem can sometimes be avoided.

**Example.** Let  $c_i$  be the value of the attribute  $i$  of case  $c$  that refers to a quality of the case (regarding our previously described READEE prototype, the quality could be, for example, the size of the on-board memory of a DSP). Let  $q_i$  denote the user's required (query) value for this attribute. A typical local similarity measure regarding the primary (quality) aspect of this attribute might follow the condition

$$\text{sim}_{\text{prim}_i}(q_i, c_i) \begin{cases} < 1 & \text{if } c_i < q_i \\ = 1 & \text{if } c_i \geq q_i \end{cases} . \quad (1)$$

This similarity measure expresses that an on-board memory size greater than or equal to the value specified in the query is preferred over smaller sizes. Figure 3 shows such a quality-oriented function.

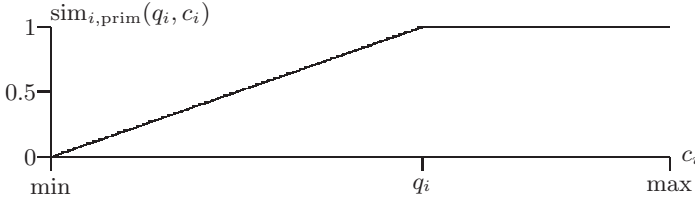


**Fig. 3.** Example similarity measure for the quality aspect of an attribute  $c_i$

To reflect the cost aspect of the same attribute, one would define another similarity measure following the condition

$$\text{sim}_{\text{cost}_i}(q_i, c_i) \begin{cases} = 1 & \text{if } c_i \leq q_i \\ < 1 & \text{if } c_i > q_i \end{cases} . \quad (2)$$

This measure follows the above stated requirement that costs may be as high as necessary to fulfill the primary quality requirements but unnecessary high costs should be avoided. Figure 4 shows such a cost sensitive function. The value  $q_i$  can be seen as a local PNC. The gradient of the decreasing part of the function indicates how sensitive the costs are to variations of the value of attribute  $i$ .

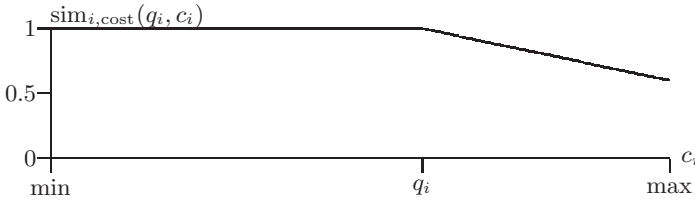


**Fig. 4.** Example similarity measure for the cost aspect of an attribute  $c_i$

To combine the local similarities it might be sufficient to declare  $\text{sim}_{\text{prim}}$  and  $\text{sim}_{\text{cost}}$  as weighted sums of their respective local similarity measures:

$$\text{sim}_{\text{prim}}(q, c) = \sum_i w_i \text{sim}_{\text{prim}_i}(q_i, c_i) \quad (3)$$

$$\text{sim}_{\text{cost}}(q, c) = \sum_i w'_i \text{sim}_{\text{cost}_i}(q_i, c_i) . \quad (4)$$

The weights  $w_i$  correspond to each attribute's contribution to the primary quality criteria, while  $w'_i$  reflect each attribute's contribution to the overall costs of the case.<sup>4</sup> If for each  $\text{sim}_{\text{cost}_i}$  the local PNC is known, the combined PNC for  $\text{sim}_{\text{cost}}$  is also implicitly determined.  $\text{sim}_{\text{prim}}$  and  $\text{sim}_{\text{cost}}$  are two conflicting similarity measures that are, however, calculated from values of the same set of attributes.

### 3.3 Combining the Models

The individual similarity measures  $\text{sim}_{\text{prim}}$  and  $\text{sim}_{\text{cost}}$  finally must be combined into one common similarity measure  $\text{sim}$ . This combined measure should have some general properties:

$$\forall q, c \quad : \quad 0 \leq \text{sim}(q, c) \leq 1 \quad (5)$$

$$\text{sim}_{\text{prim}}(q, c) = 1 \wedge \text{sim}_{\text{cost}}(q, c) = 1 \quad \Rightarrow \quad \text{sim}(q, c) = 1 \quad (6)$$

$$\text{sim}_{\text{prim}}(q, c) = 0 \quad \Rightarrow \quad \text{sim}(q, c) = 0 . \quad (7)$$

Equation (7) underlines the importance (but not necessarily dominance) of the primary similarity measure. In addition to these properties it is desirable to have the ability to adjust the cost sensitivity of  $\text{sim}$  to the requirements of the application or the preferences of the user.

Given that (5) holds for both  $\text{sim}_{\text{prim}}$  and  $\text{sim}_{\text{cost}}$  the following definition of  $\text{sim}$  accounts for (6) and (7):

$$\begin{aligned} \text{sim}(q, c) &= \text{sim}_{\text{prim}}(q, c) \text{sim}_{\text{cost}}^\omega(q, c) \\ &\text{with } \omega \in \mathbb{R}^+ \setminus \{0\} . \end{aligned} \quad (8)$$

There are other possibilities to combine  $\text{sim}_{\text{prim}}$  and  $\text{sim}_{\text{cost}}$ . Equation (8) is only one possibility but it is easy to compute and the parameter  $w$  provides an easy means for accounting for varying, user-dependent cost sensitivity.

Please note that using a weighted sum instead of a product would violate condition (7). Also note that it is not feasible to first combine the local similarity measures  $\text{sim}_{\text{prim}_i}$  and  $\text{sim}_{\text{cost}_i}$  to local measures  $\text{sim}_i$  and then combine the  $\text{sim}_i$  with a weighted sum to a global similarity. This is because the weights  $w_i$  and  $w'_i$  from (3) and (4) are different. The different weights account for the fact that certain attributes may be very costly but contribute only a small amount to quality, and vice versa.

<sup>4</sup> It is worth noting that the weights for  $\text{sim}_{\text{cost}_i}$  are usually constants while the weights for  $\text{sim}_{\text{prim}_i}$  often depend on the query or on the user's individual preferences.

### 3.4 Completing Undefined Attributes

The biggest problem in finding the global PNC are undefined values of query attributes<sup>5</sup> because for these attributes the local PNC is not known.

There is a significant difference in handling undefined query values for the primary and secondary similarity measure: in the selection support domain from our example, undefined query values are usually interpreted as “don’t care” specifications. I. e. there are no special requirements for these attributes to fulfill the primary quality criteria. However, these attributes may still contribute to the secondary cost criteria.

Because of the required property 3b it is important to find a suitable definition of  $\text{sim}_{\text{cost}_i}(q_i, c_i)$  for the case that  $q_i$  is a “don’t care” specification. For this case, the similarity measure needs to know the attribute’s contribution to the overall cost that is “necessary” to fulfill the primary requirements. This minimal cost contribution could be calculated if we could deduce a minimal requirement  $q_{\min_i}$  for attribute  $i$  from the query values of the other attributes.

This, of course, is not possible. However, we can try to approach this problem in a case-based way: assuming that the case base is a representative profile of all designs that “make sense”, we can make an assumption about  $q_{\min_i}$  by the following argument:

1. If there exists a set  $C$  of cases that truly fulfill all primary requirements (i. e.  $\text{sim}_{\text{prim}_j}(q_j, c_j) = 1$  for  $c \in C$  and  $q_j$  defined) then  $q_{\min_i} \leq \min\{c_i : c \in C\}$  .
2. Trusting in our case base we expect that most primary requirements are at least *individually* truly fulfilled by some cases. From the set  $C'$  of all cases that at least fulfill one primary requirement we expect  $\min\{c_i : c \in C'\}$  to be a lower bound for  $q_{\min_i}$  .

With these assumptions (whose reasonability, of course, depends on the application and on the case base) we can complete the query and estimate values for undefined attributes  $q_i$  for the purpose of calculating the local PNC and therefore  $\text{sim}_{\text{cost}_i}$ . However, since  $q_{\min_i}$  cannot be exactly determined (they will probably be too low in most cases), the required property 3b can only be roughly approximated. As we argued before, using too low a value for the PNC will prevent the overall similarity from having a plateau of indifference.

### 3.5 Defining Secondary Measures Relatively

In Sect. 3.2 we assumed that the costs can be expressed as a combination of contributions from a number of independent attributes. Independence of similarities of attributes is an assumption that is made very often in CBR systems. One argument for this is that if the attributes’ similarities are not independent,

---

<sup>5</sup> Undefined case attributes are not so common in selection support domains that we are mostly concerned of and are therefore not covered here.

they have been chosen without care and the vocabulary [7] of the case base must be changed appropriately. – Of course, it is often not that simple.

Even if the vocabulary has been chosen carefully and the local attribute similarities are independent regarding the primary similarity measure, they can generally not be expected to be independent for the secondary similarity measure, as well. This limits the applicability of (4). However, the exact calculation scheme for  $\text{sim}_{\text{cost}_i}$  is not important as long as the point of necessary cost can be estimated to a suitable degree of accuracy.

A straight forward way to eliminate all problems concerning independence of attributes is to calculate  $\text{sim}_{\text{cost}_i}$  based on a single value only. Although the real costs of an IP can not be determined in an easy way, it is still possible for an expert to arrange the IPs relative to each other according to their expected costs. For this purpose we introduce a cost-related attribute whose values are normalized within a given range. The cheapest and most expensive IPs are assigned the minimum and maximum value from this range, respectively.

Figure 5 shows a prototypical implementation of an editor for such an attribute. In this screen shot, the slider position indicates the relative cost of the currently edited case. It is visually set in relation to the costs of the other cases (only user-selectable “landmark cases” are shown). The buttons at the left and right of the slider allow the specification of costs that are outside of the range of currently available cases. This leads to a re-calculation of the cost values of all cases in order to fit into the given range of  $[0, 1]$ . The “Normalize” function scales all values to optimally occupy this range.

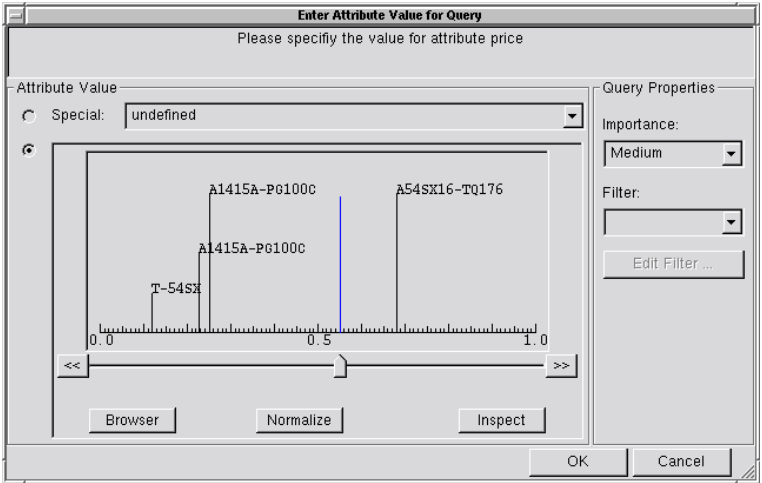


**Fig. 5.** Prototypical attribute editor for defining costs of a case relative to the costs of other cases

For this attribute we define a similarity measure similar to that of Fig. 4. To estimate the point of necessary cost we can apply an algorithm similar to the one described in Sect. 3.4:

1. If there exists a set  $C$  of cases that truly fulfill all primary requirements (i.e.  $\text{sim}_{\text{prim}_j}(q_j, c_j) = 1$  for  $c \in C$  and  $q_j$  defined) then  $\text{cost}_{\min} \leq \min\{\text{cost}(c) : c \in C\}$ .
2. From the set  $C'$  of all cases that at least fulfill one primary requirement we expect  $\min\{\text{cost}(c) : c \in C'\}$  to be a lower bound for  $\text{cost}_{\min}$ .

This algorithm might be enhanced for individual applications. For example, it might be possible to alter the conditions for  $C'$  so that only cases are included that fulfill a minimum number of primary requirements. It might also be sensible to explicitly exclude certain requirements, that are almost always fulfilled, from this condition.

The drawback of this approach is that an extra attribute is needed whose value has to be provided and maintained. However, the editor is designed to help in maintaining the values of the other cases as good as possible. The estimation of the PNC is possibly less accurate because there is no definite information that can be relied on. On the other hand, if the end-user can be expected to be familiar with the costs of some “landmark cases”, the ability to formulate the query relative to other known objects might make it feasible to ask the user to provide a value for the PNC.

## 4 Conclusion

We have suggested concepts for modeling vague and implicit knowledge about qualitative criteria, especially about costs in certain selection support applications. We have presented two different approaches: the first one using a somewhat artificial explicit representation as it has already been used in some CBR systems. This is most appropriate for qualitative criteria which are nonetheless explicitly determined by the user (or the application), and do not depend on other criteria. The second approach makes it possible to formulate qualitative criteria without any kind of quantification but depending on another primary criteria.

The main strengths of this approach lie in its ability to model implicit or “secondary” criteria which heavily depend on the fulfillment of a primary criterion. This can occur in pre-sales support applications (see the family car example from the introduction) as well as in a number of other decision support systems. The main motivation originated from a reuse assistant for electronic designs.

The main difficulty when using this approach for handling cost criteria is to find a good estimation of the minimum “necessary” requirements of unspecified query attributes. This difficulty also arises for other kinds of secondary similarity measures that require for something different than a “don’t care” semantics for undefined attributes. This is true for costs, since they are influenced by all properties of a case, whether they are requested or not. There is some more



research needed to find good heuristics for estimating good values for  $q_{\min_i}$  but this can only be done in a more application specific context.

The idea of having a primary and a secondary similarity measure can be extended to having more than one secondary similarity measure ( $\text{sim}_{\text{sec}_i}$ ,  $i = 1, \dots, n$ ). One straight-forward extension of (8) would be:

$$\text{sim}(q, c) = \text{sim}_{\text{prim}}(q, c) \prod_{i=1}^n \text{sim}_{\text{sec}_i}^{\omega_i}(q, c) \quad (9)$$

but the relative importance of  $\text{sim}_{\text{prim}}$  (c.f. (7)) also allows for

$$\text{sim}(q, c) = \text{sim}_{\text{prim}}(q, c) \left( \sum_{i=1}^n \omega_i \text{sim}_{\text{sec}_i}(q, c) \right)^{\omega} \quad (10)$$

Again,  $\omega_i$  and  $\omega$  can be used to adjust the sensitivity of the overall similarity to each of the secondary aspects of similarity. These two alternatives (and possibly others) need further evaluation in the context of specific applications.

As the concepts described in Sect. 2 have been implemented in the READEE prototype for DSP selection support, the approach from Sect. 3 is being employed for the implementation of the second prototype of the READEE project. This application is going to be a selection support system for field programmable gate arrays (FPGA). It is work in progress.

## References

1. Ivo Vollrath and Peter Oehler. Intelligent retrieval of electronic designs in online catalogs. In Wolfgang Wilke and Jürgen Schumacher, editors, *KI-98 Workshop on Intelligent Systems and Electronic Commerce*, number LSA-98-03E in LSA Report, pages 57–63, Kaiserslautern, 1998. Centre for Learning Systems and Applications (LSA). 309
2. Norbert Faulhaber and Ralf Seepold. An efficient similarity metric for IP reuse in RMS. In Kunzmann and Seepold [8], pages 23–32. 309
3. Ranto Rafidinitrimo, Philippe Coeurdevey, and Gabrièle Saucier. IP key features and an object oriented IP catalog management system. Design & Reuse Technical Paper, Grenoble, France, On-Line, Last seen at [http://www.design-reuse.com/NEWS/paper02\\_98.html](http://www.design-reuse.com/NEWS/paper02_98.html), February 1999. 309
4. Michael M. Richter and Klaus-Dieter Althoff. Similarity and utility in non-numerical domains. In *Mathematische Methoden der Wirtschaftswissenschaften*, pages 403–413. Physika-Verlag, 1999. Last seen at <http://www.cbr-web.org/documents/RichterSimilarity99.pdf>. 311
5. Ivo Vollrath. Reuse of complex electronic designs: Requirements analysis for a CBR application. In Barry Smyth and Pádraig Cunningham, editors, *Advances in Case-Based Reasoning: 4th European Workshop, proceedings/EWCBR-98*, volume 1488: Lecture notes in artificial intelligence of *Lecture notes in computer science*, Berlin, 1988. Springer. 311
6. Peter Oehler and Ivo Vollrath. Are you READEE for IPs? In Kunzmann and Seepold [8]. 311

7. Michael M. Richter. The knowledge contained in similarity measures. Invited Talk at the *International Conference on Case-Based Reasoning (ICCBR-95)*, 1995. Last seen at <http://www.cbr-web.org/documents/Richtericcbr95remarks.html>. 318
8. Arno Kunzmann and Ralf Seepold, editors. *Proceedings 2. GI/ITG/GMM Workshop: Reuse Techniques for VLSI Design*, FZI-Report 3-13-9/98, Karlsruhe, September 1998. Forschungszentrum Informatik. 320

# Active Delivery for Lessons Learned Systems

Rosina Weber<sup>1,2</sup>, David W. Aha<sup>2</sup>, Hector Muñoz-Quila<sup>2,3</sup>, and Leonard A. Breslow<sup>2</sup>

<sup>1</sup>Department of Computer Science, University of Wyoming  
Laramie, WY 82071-3682

<sup>2</sup>Navy Center for Applied Research in Artificial Intelligence  
Naval Research Laboratory, Washington, DC 20375

<sup>3</sup>Department of Computer Science, University of Maryland  
College Park, MD 20742-3255  
lastname@aic.nrl.navy.mil

**Abstract.** Lessons learned processes, and software systems that support them, have been developed by many organizations (e.g., all USA military branches, NASA, several Department of Energy organizations, the Construction Industry Institute). Their purpose is to promote the dissemination of knowledge gained from the experiences of an organization's employees. Unfortunately, lessons learned systems are usually ineffective because they invariably *introduce* new processes when, instead, they should be *embedded* into the processes that they are meant to improve. We developed an embedded case-based approach for lesson dissemination and reuse that brings lessons to the attention of users rather than requiring them to fetch lessons from a standalone software tool. We demonstrate this active lessons delivery architecture in the context of HICAP, a decision support tool for plan authoring. We also show the potential of active lessons delivery to increase plan quality for a new travel domain.

## 1 Introduction

*Lessons learned* (LL) are validated working knowledge derived from successes or failures that, when reused, can significantly impact an organization's processes (Secchi, 1999). *LL processes* refer to an organization's efforts for managing lessons learned. Hundreds of organizations have or plan to develop LL processes with the goal of enhancing job safety, reducing costs, improving quality, and/or increasing problem-solving speed.

*LL systems* (i.e., software for supporting LL processes) have been implemented in military organizations since the mid 1980s and in government and commercial organizations during the 1990s as knowledge management (KM) (Davenport & Prusak, 1998) solutions for documenting tacit lessons whose reusable content can benefit targeted organizational processes. Surveyed organizations typically address five LL sub-processes: collect, verify, store, disseminate, and reuse (Weber et al.,

2000a). LL systems have been used to support collection (e.g., online lesson submission forms) and dissemination (e.g., standalone retrieval tools). Unfortunately, standalone dissemination systems do not reflect a central KM tenet: *to be successful, knowledge dissemination methods must be embedded in the processes they are intended to support* (Reimer, 1998; Aha et al. 1999; Leake et al., 2000). Thus, they are typically underutilized and do not promote knowledge sharing and reuse.

Previously deployed LL systems for lesson dissemination have not used AI approaches, except for RECALL (Sary and Mackey, 1995), which uses conversational case retrieval to retrieve lessons. However, RECALL did not employ a knowledge representation that was optimized to promote lesson reuse, and it is a standalone LL tool. We address issues concerning lesson representation in (Weber et al., 2000c). This paper addresses how AI approaches can be used to develop embedded software architectures for LL systems. In particular, we introduce an embedded case-based architecture for LL *dissemination* and *reuse* in HICAP (Muñoz-Avila et al., 1999), a multi-modal plan authoring tool, and demonstrate its utility in an evaluation on a new travel planning domain. We begin by summarizing related work in Section 2 and introducing LL processes and systems in Section 3. Section 4 then describes HICAP and introduces our embedded active lessons delivery module. Section 5 describes an initial empirical study. Finally, we discuss implications and future needs in Section 6.

## 2 Related Work

The most ambitious investigation of LL processes was performed by the Construction Industry Institute's (CII's) Modeling Lessons Learned Research Team (Fisher et al., 1998). They surveyed 2400 organizations, characterized the 145 initial responses as describing 50 distinct LL processes, and performed follow-up, detailed investigations with 25 organizations. They found strong evidence that most organizations were using insufficient dissemination processes.

Secchi et al. (1999) found that only 4 of the 40 organizations who responded to their survey used software to support their LL process. In both surveys, none of the responding organizations implemented an active LL process for lesson dissemination, probably because software was not used to control the process(es) targeted by the lessons, or elicited lessons were immediately/manually incorporated into the targeted process (e.g., into the organization's best practice manuals, or by requiring project members to read through project-relevant lessons prior to initiating a new project).

Similarly, contributions at previous workshops on LL processes (SELLS, 1999; Secchi, 1999) were limited to standalone systems, and few authors have discussed using artificial intelligence (AI) to assist LL processes (e.g., Vandeville and Shaikh (1999) briefly mention using fuzzy set theory to analyze elicited lessons). The AAAI 2000 Workshop on *Intelligent Lessons Learned Systems* (Aha & Weber, 2000) will be the first workshop to specifically promote AI contributions to this topic.

We concluded from our own survey (Weber et al., 2000a)<sup>1</sup> that no deployed LL system uses an embedded architecture for lesson dissemination, and only three such

---

<sup>1</sup> Please see [www.aic.nrl.navy.mil/~aha/lessons](http://www.aic.nrl.navy.mil/~aha/lessons) for pointers to on-line lessons learned systems.

architectures for LL and related systems have been proposed: ACPA (Johnson et al., 2000), ALDS (Weber et al., 2000f), and CALVIN (Leake et al., 2000). These architectures each employ case retrieval. However, ACPA uses a form of task-model tracking to prompt users with corporate stories, and CALVIN collects task-specific lessons for searching information sources. In contrast, ALDS focuses on disseminating and reusing organization-specific lessons for multiple tasks in planning domains, and is the first LL dissemination architecture that supports executable lesson reuse. This paper discusses the implementation and initial analysis of ALDS in HICAP.

In planning contexts, LL dissemination sub-processes can be used to adapt plans by applying lessons. This is related to using cases to guide case adaptation (Leake & Kinley, 1998). Previous such approaches explicitly recorded case adaptation cases and made them available in subsequent adaptation episodes. In our work, lessons are first recorded from experiences in executing plans (e.g., in military operations), then are checked in a systematic validation process by LL experts, and those that are accepted are finally made available for modifying future planning efforts. Therefore, unlike adaptation cases, lessons are not restricted to modifying case solutions, but instead are intended to modify existing organizational (e.g., planning) processes. Also, our approach is immersed in a decision support environment, in which users can optionally ignore or amend a lesson's recommendation.

### 3 Lessons, Processes, and Systems

The following definition has been adopted by NASA and European Space Agencies:

*"A **lesson learned** is a knowledge or understanding gained by experience. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure. Successes are also considered sources of lessons learned. A lesson must be significant in that it has a real or assumed impact on operations; valid in that is factually and technically correct; and applicable in that it identifies a specific design, process, or decision that reduces or eliminates the potential for failures and mishaps, or reinforces a positive result."* (Secchi et al., 1999)

This definition highlights several criteria for lessons learned: they are knowledge artifacts, obtained from either positive or negative experiences, are validated for correctness, and, when reused, can significantly impact an organization's processes.

A **LL process** implements a strategy to collect, verify, store, disseminate, and reuse lessons to continually support an organization's goals. Thus, it defines how to use *tacit* experiential knowledge in an organization's activities, capturing the experiential knowledge from employees whose knowledge might be lost when they leave the company, shift projects, retire, or otherwise become unavailable. LL processes typically target decision-making or execution processes for various types of user groups (i.e., managerial, technical) and organizations (e.g., commercial, military). In this paper, we focus on managing lessons to support planning processes.

Flowcharts describing LL processes abound; organizations produce them to communicate how lessons are to be acquired, verified, and disseminated (SELLS,

1999; Fisher et al., 1998; Secchi, 1999). Figure 1 displays a typical LL process, composed of five sub-processes (i.e., collect, verify, store, disseminate, and reuse), where reuse does not take place in the same environment as the other sub-processes.

Existing, deployed *LL systems* do not support all LL processes. Organizations typically do not develop software to support verification or reuse. Instead, they use electronic submission forms to facilitate lesson collection, and use a standalone tool to support lesson dissemination (Weber et al., 2000a). Users interacting with this standalone tool are expected to browse the stored lessons, studying some that can assist them with their decision-making process(es). However, based on our interviews and discussions with members of several LL organizations (e.g., at the Joint Warfighting Center, the Department of Energy, NASA's Goddard Space Flight Center, the Navy Facilities Engineering Command, CII), and many intended users of LL systems, we concluded that users do not use available standalone LL systems, which are usually ineffective because they force users to master a *separate* process from the one they are addressing, and impose the following unrealistic assumptions:

1. Users are convinced that using LL systems is beneficial, and can find them.
2. Users have the time and skills to successfully retrieve relevant lessons.
3. Users can correctly interpret retrieved lessons and apply them successfully.
4. Users are reminded of the potential utility of lessons when needed.

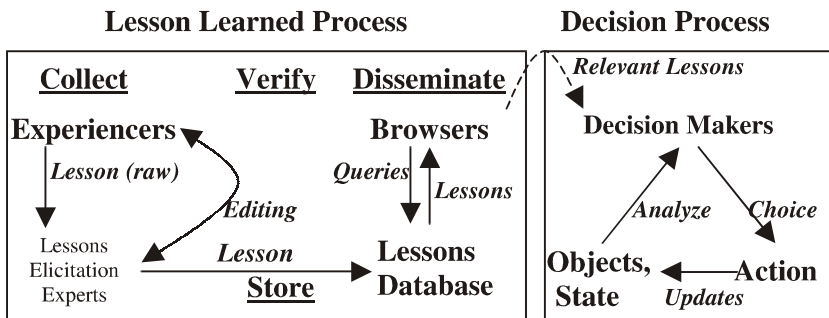


**Fig. 1.** Most lessons learned process are separated from the decision processes they support

This motivated us to develop an *active lessons delivery* architecture for lesson dissemination (Figure 2). In this approach, reuse occurs in the same environment as other sub-processes; the decision process and lessons learned process are in the same context. This embedded architecture has the following characteristics/implications:

1. The LL process interacts directly with the targeted decision-making processes, and users do not need to know that the LL module exists nor learn how to use it.
2. Users perform or plan their decision-making process using a software tool.
3. Lessons are brought to the user's attention by an embedded LL module in the decision-making environment of the user's decision support tool.
4. A lesson is suggested to the user only if it is applicable to the user's current decision-making task and if its conditions are similar to the current conditions.
5. The lesson may be applied automatically to the targeted process.

This process shifts the burden of lesson dissemination from a user to the software, but requires intelligent software modules and systematic knowledge engineering efforts.

## 4 A Case-Based Approach for Active Lessons Delivery

This section details a case-based approach for active lessons delivery, which we have implemented in a module of HICAP (Muñoz-Avila et al., 1999). Sections 4.1 and 4.2 detail HICAP and its active lessons delivery module, respectively.

### 4.1 Plan Authoring in HICAP

HICAP (Hierarchical Interactive Case-based Architecture for Planning) is a multi-modal reasoning system that helps users to formulate a task hierarchy, which is represented as a triple  $H = \{T, <, ^\wedge\}$ , where each task  $t \in T$  is defined by its name  $t_n$ ,  $<$  defines a (partial) ordering relation on tasks, and  $t_1 \wedge t_2$  means that  $t_1$  is a parent of  $t_2$  in  $T$ . Task hierarchies are created in the context of a state  $S = \{<q, a>^+\}$ , represented as a set of question/answer pairs.

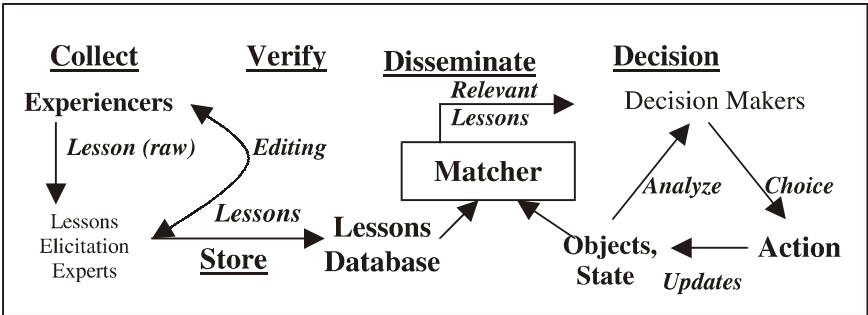


Fig. 2. Active lessons delivery for lesson dissemination integrated with the decision process

Although HICAP manipulates several objects (e.g., resources) and decisions (e.g., resolve a resource conflict), we focus this first implementation of ALDS on task decomposition. HICAP provides users with three ways to decompose tasks into subtasks. First, it supports manual task decomposition. Second, users can select a task  $t$  to decompose and use HICAP's conversational case retriever (NaCoDAE/HTN) to *interactively* select a stored task decomposition for  $t$ , assuming that specific cases exist for decomposing  $t$ . This involves a conversational interface, where the user sees displays of top-ranking solutions and unanswered questions, can iteratively answer any of the displayed questions (which adds a  $<q, a>$  pair to  $S$  and updates the two displays), and can end the conversation by selecting a displayed solution (i.e., a task decomposition). Third, users can select a generative planner (SHOP) to *automatically* decompose  $t$ , assuming methods or operators exist for decomposing  $t$ . We describe

the tight integration of SHOP with NaCoDAE/HTN, in the SiN algorithm, in (Muñoz-Avila et al., 2000).

## 4.2 A Case-Based Active Lessons Delivery Module

Planning tasks, even everyday ones like those for planning how to travel between two locations, can involve several decisions whose effect on plan performance variables (e.g., time, cost) depends on a variety of state variables (e.g., weather, amount of luggage). Without a complete domain theory, HICAP cannot guarantee it will produce a correct plan for all possible states. But obtaining a complete domain theory is often difficult, if not impossible. Besides representing typical experiential knowledge, lessons can help fill gaps in a domain theory so that, when reused appropriately during planning, they can improve plan performance. This is the motivation for applying lessons while using HICAP.

Figure 3 summarizes the behavior of the active lessons delivery module, which monitors task selections, decompositions, and state conditions to assess similarities between them and the stored lessons. When a stored lesson's applicable decision matches the current decision and its conditions are a good match with the current state, then the lesson is brought to the user's attention to influence decision making. When a user implements a prompted lesson's task decomposition (i.e., reusing the lesson), the current task hierarchy is modified appropriately.

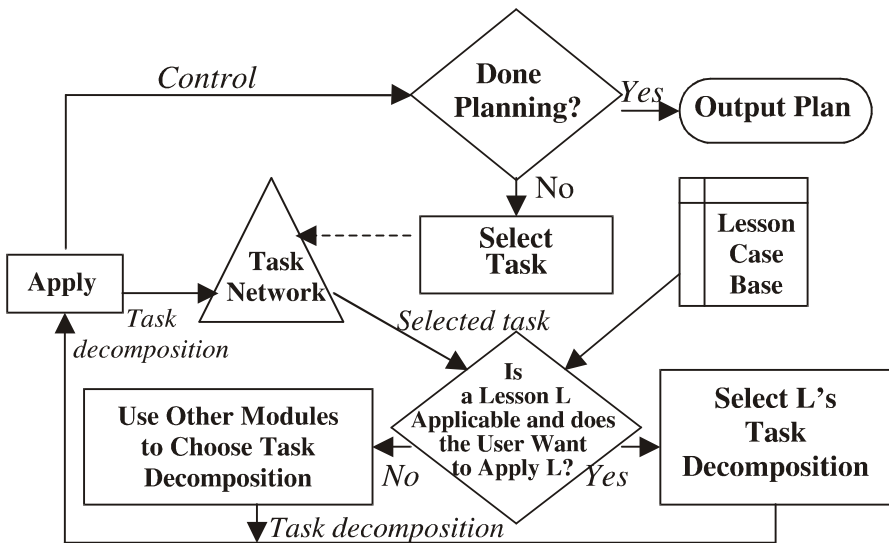


Fig. 3. HICAP's lessons delivery sub-process during plan elaboration

This first implementation of ALDS incorporates a simple strategy. First, instead of representing lessons using all six of the characteristics of an idealized lesson representation (Weber et al., 2000c), we instead borrowed the representation used by NaCoDAE/HTN for task decomposition cases. That is, a lesson is indexed by the *originating task* that it can decompose and a set of <question, answer> pairs defining



its *conditions*, and contains a *suggestion* (e.g., a task decomposition). Thus, if the conditions are a good match to the current planning state, then the user should consider decomposing the current task into the lesson's suggested subtasks.

Second, we borrowed NaCoDAE/HTN's similarity function for cases, and used a thresholded version to determine when a lesson should be prompted to a user.

Finally, we only supported one of many possible lesson suggestions (i.e., task decomposition). In future implementations, lessons will be able to represent suggestions that, when applied, will not be limited to task substitution. For example, a lesson might suggest a task decomposition, or using an alternative resource assignment for a given task, recommend changing some temporal orderings of tasks, or suggest other edits to any of the objects used by HICAP to define plans.

## 5 Empirical Study

Our hypothesis is that the ALDS architecture is superior to the traditional standalone architecture for supporting lesson dissemination in planning tasks. That is, we claim our architecture is better suited for promoting lesson reuse. This hypothesis is difficult to evaluate; it requires evaluating the plans created by operational users who use these two approaches in repeated planning tasks. Dependent variables would primarily include agreed-upon measures of plan quality, which would be planning domain dependent. Unfortunately, at this time, we do not have access to sufficient users nor realistic plan evaluation simulators (e.g., although our group used ModSAF to evaluate plans generated by HICAP for noncombatant evacuation operations (NEOs) in (Muñoz-Avila et al., 1999), the simulator could only be used for a single subtask of an entire NEO plan.). Therefore, we instead focused our experiment on simulated users in a novel travel planning domain for which we could create an adequate simulator (Section 5.1). Also, because it's difficult to simulate how a user might benefit from a standalone lesson dissemination tool, we instead compared plan generation when using the active lessons delivery module vs. not using it (Section 5.2), where our hypothesis is that *using lessons will improve plan quality*.

### 5.1 Personal Travel Domain

This domain was inspired by DARPA's Active Templates program, which concerns (in part) the development of inferencing tools for travel plan generation. We developed a knowledge base and plan evaluator for this domain, in which the objective was to create a plan to travel from one of ten locations in the Washington, DC metropolitan area (e.g., Adam's Morgan, Bethesda, DuPont Circle) to downtown New York City. We represented seven transportation methods, including four inter-city and three intra-city methods: {airplane, city bus, inter-city bus, shuttle van, subway, taxi, train}. Plans have 3-5 segments, while plan hierarchies have exactly four levels. This defines a space of approximately 756 possible plans. A travel *state* is defined by up to eight variables, represented as questions (Table 1).

Table 1. Questions used to define personal travel states

| Question                                            | Possible Answers    |
|-----------------------------------------------------|---------------------|
| Is it raining hard?                                 | Y/N                 |
| Is there any major accident or unusual event?       | Y/N                 |
| What is the chance of large snow accumulation?      | High, moderate, low |
| How many passengers?                                | 1,2,3,3+            |
| Is there any luggage to check?                      | Y/N                 |
| Are any children or seniors involved?               | Y/N                 |
| Is travel occurring on a holiday? If so, which one? | Holiday name, or No |
| What is the starting day of the week for this trip? | 7 distinct answers  |

Our hand-coded knowledge base consists of ten methods and one (movement) operator for HICAP’s JSHOP module, and 40 cases for NaCoDAE/HTN. Each plan’s task decompositions were obtained by applying 4-6 methods, 3-5 operators (i.e., the same operator 3-5 times), and 3-5 cases. An example plan is: take(taxi,Bethesda, BWI), take(plane, BWI, JFK), take(shuttle, JFK, Downtown NYC).

|                                                                                                                                                                  |                                                                                                                                                                          |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Head</b><br>travelCity(Penn Station,<br>DowntownNYC)<br><b>Questions</b><br>Is it raining hard? <b>No</b><br>How many passengers? <b>2</b><br><b>Subtasks</b> | <b>Head</b><br>travelCity (taxi, Penn Station,<br>DowntownNYC)<br><b>Questions</b><br>Is it raining hard? <b>Yes</b><br>How many passengers? <b>2</b><br><b>Subtasks</b> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Fig. 4. Example case (left) and lesson (right) from the personal travel domain

The version of HICAP used in this paper is deterministic; given a state and a top-level goal (i.e., start and destination locations), it will always generate the same plan. However, our personal travel plan evaluator is non-deterministic; it can generate a different time duration for each of a given plan’s segments each time the plan is executed. The evaluator is given a plan and the current world state. For each run, it outputs whether the plan succeeded and, if it did succeed, how long the trip took as well as its cost. We relied on interviews with domain experts for information on costs and expected time duration. A plan will fail if either a huge delay occurs (e.g., due to an unusual event) that causes a plan cancellation, or when segment delays cause a late arrival for a segment requiring a fixed time departure (e.g., an airplane flight). For each segment, we applied an exponential delay function that is influenced by world state conditions. For example, a flight segment will incur a longer delay for higher chances of large snow accumulation, especially on holidays (i.e., high travel days). Segments are categorized into short, medium, and long lengths, and delays can range from 0 up to 4.5 times a segment’s anticipated duration before a plan is cancelled.

Figure 4 depicts an example case from the personal travel domain and a lesson that modifies it. In the case, a taxi is taken from Penn Station to downtown New York City when there is no rain and the number of passengers is 2. The lesson modifies this case by indicating that, when it is raining, the subway should be taken instead. (It is well known that it is almost impossible to find a cab when it rains in New York City.)

## 5.2 Experiments: Methodology and Results

We explored whether a case-based active lessons delivery strategy for this travel domain can improve plan performance, as measured by success frequency and mean time duration, without increasing travel cost.

We generated lessons by analyzing some problem results. We define a *goal* to be a pair of <start, destination> locations and a *problem* as a pair of <goal, world state>. We selected ten goals, corresponding to the ten ways in which travel could take place between Washington, DC and NYC. For each goal, we generated 10 random world states (i.e., providing answers to *all* 8 questions), thus yielding 100 total problems. HICAP was then used to generate a plan for each problem. Then, for each of the 100 <plan, world state> pairs, the plan evaluator was run 10 times. For each run, we measured whether the plan executed to completion, the plan's (simulated) duration, and the number of lessons used to create it.

We then used the following three manual procedures, five times each, to create lessons. These procedures were designed to reduce human biases in lesson creation.

1. *Plan failure lessons*: Select a problem  $g_p$  (with plan  $p$ ) whose success rate was lowest for a randomly selected goal  $g$ 's problems. Then try to reduce its trip duration as follows. Locate  $p$ 's segment  $p_f$  where failure most frequently occurred, and replace a segment preceding it with a faster transportation method. If this increased trip cost more than 15%, then also replace the transportation method of a segment in  $p$  following  $p_f$  with a cheaper transportation method. However, if this could not account for the cost increase, then abandon trying to reduce trip duration and instead simply start plan  $p$  earlier by the mean time that the traveler was late for  $p_f$ .
2. *Positive (short) duration lessons*: Randomly select a goal  $g$  and its subset of problems  $P$  whose success rates were at least 50%. Select problems  $g_{short}$  and  $g_{ave}$  from  $P$  whose mean duration were shortest and most similar to the average, respectively. Let  $t_{short}$  and  $t_{ave}$  be the starting times of the inter-city plan segments  $p_{short}$  and  $p_{ave}$  in the plans of these two problems. Let  $\Delta = t_{ave} - t_{short}$ . If the mean time that the traveler waited for  $p_{ave}$ , minus the s.d. for this problem, is greater than  $\Delta$ , then substitute  $p_{short}$  for  $p_{ave}$  in the plan for  $g_{ave}$ .
3. *Negative (long) duration lessons*: Same as (3), but focusing on  $g_{ave}$  and  $g_{long}$ . Figure 5 displays one of the *plan duration* lessons.

Each lesson was encoded with 8 <question, answer> pairs, corresponding to the world state during the experience from which it was derived. These procedures produced 15 lessons. We tested HICAP's performance with vs. without lessons by examining whether the lessons could improve plan performance on the 100 problems from which they were derived.

LESSON: lessontravelWASNYPtrain1230lesson

NAME

lessontravelWASNYPtrain1230lesson

TASK

WAS

NYP

train

1230

TEXT

PRECONDITIONS

| Question                                             | Answer |
|------------------------------------------------------|--------|
| travelingWithChildren Is the traveler travelling ... | Yes    |
| heavyRain IS there a heavy rain?                     | No     |
| heavySnow Are there good chances of snowing o...     | No     |

SUGGESTIONS

NEW TASKS:

WAS NYP train 1200

TASK ORDERINGS: None

Apply

Quit

Quit All

Fig. 5. Lesson suggesting to take an earlier train to reduce trip duration

We implemented a simulated HICAP user that uses the process shown in Figure 3 and the following behavior during NaCoDAE/HTN conversations:

- It always answers the top-ranking displayed question for which it has an answer.
- It answered questions until either none remained unanswered or one of the solutions exceeded a retrieval threshold, which we set to 50%.

The results of this experiment are summarized in Table 2. In the training set, plans generated by HICAP with the case-based active lessons delivery module reduced average trip duration by over 30 minutes at the same price level, and even improved the success rate. This improvement demonstrates the potential benefit of lesson reuse because it changed the relation between trip duration and success rate (*i.e.*, we always increase the chance of success if we depart earlier).

These results suggest that the active lessons delivery approach can potentially generate better plans for realistic problem domains (e.g., planning for NEO operations). Similar improvements may yield improvements in plans for domains where safety issues and speed are paramount to success.

Table 2. Experimental results with the 100 problems

| #Less<br>ons | Succe<br>ss Rate | Mean (s.d.) Trip<br>Duration | Lesso<br>n use rate | Average<br>Price |
|--------------|------------------|------------------------------|---------------------|------------------|
| 0            | 70%              | 9h45min<br>(3h14min)         | 0%                  | \$ 221           |
| 20           | 71%              | 9h14min<br>(3h35min)         | 31%                 | \$ 220           |

## 6 Discussion

Our evaluation of the active lessons delivery module demonstrates how this approach, when embedded in a decision-making (i.e., planning) process, can improve the results of that process. Although we used simulated users in our experiments to reduce human biases during the evaluation, we stress that this is a mixed-initiative approach in which humans interact with HICAP to generate plans. Perhaps the most unique aspect of our approach, ALDS, is that it allows users to execute a lesson's suggestion (i.e., here, a task decomposition), rather than limit them to browsing the suggestion.

We are currently investigating the lesson selection heuristics and their capability in generating useful lessons. In future research, we intend to explore the utility of lessons that are not restricted to task decomposition recommendations. We will also return to the NEO planning and/or alternative military planning domains as we investigate the utility of lessons in more applied domains.

Conceptually, HICAP's NaCoDAE/HTN module uses cases that represent task decompositions corresponding to either (generalized) standard operating procedures or decompositions that were derived from executing plans. These cases are limited to one purpose: updating the task hierarchy. In contrast, the ALDS module in HICAP uses lessons that capture experiences that, if reused, can significantly impact the performance of subsequent plans. Importantly, lessons are not limited to representing task decompositions, but can be used to apply edits to any of HICAP's objects (e.g., resource assignments, resources, task durations). In summary, although we have implemented lessons as cases in this implementation of ALDS, it was only for convenience; lessons have a more general scope than task decomposition.

## 7 Conclusion

We introduced a case-based active lessons delivery approach for a lessons learned dissemination sub-process. In comparison to traditional standalone systems, active delivery systems bring lessons to a user's attention rather than requiring them to consult a separate retrieval process, which can be problematic. In the context of a HICAP module, we showed that, for a personal travel domain, using lessons can increase plan quality (e.g., reduce mean travel duration with the same level of cost and success).

Our research goals include developing a more mature version of this module and evaluating it on a more realistic planning domain (e.g., noncombatant evacuation operations). We are currently pursuing potential transitions of HICAP, including its active lessons delivery module, to deployed (e.g., military) systems, and have begun working with the Joint Warfighting Center towards this goal. We have also begun to develop strategies for systems that use HICAP to assist with lesson collection, and plan to evaluate these strategies in our future efforts.

## Acknowledgements

This research was supported by grants from the Office of Naval Research and the Naval Research Laboratory. The travel domain was inspired by Maj. Doug Dyer's presentations for the Active Templates Program. Finally, thanks to Karen Spearman for assisting with developing the travel domain, and for our reviewers's suggestions.

## References

1. Aha, D.W., Becerra-Fernandez, I., Maurer, F., and Muñoz-Avila, H. (Eds.). 1999. Exploring Synergies of Knowledge Management and Case-Based Reasoning: Papers from the AAAI 1999 Workshop (Technical Report WS-99-10). Menlo Park, CA: AAAI Press.
2. Aha, D.W., & Weber, R. (2000). *Intelligent Lessons Learned Systems: Papers from the AAAI Workshop* (Technical Report WS-00-08). Menlo Park, CA: AAAI Press.
3. Davenport, T. H., & Prusak, L. (1998). *Working knowledge: How organizations manage what they know*. Boston, MA: Harvard Business School Press.
4. Fisher, D., Deshpande, S., & Livingston, J. (1998). *Modeling the lessons learned process* (Research Report 123-11). Albuquerque, NM: The University of New Mexico, Department of Civil Engineering.
5. Johnson, C., Birnbaum, L., Bareiss, R., & Hinrichs, T. (2000). War Stories: Harnessing Organizational Memories to Support Task Performance. *Intelligence*, 11(1), 17-31.
6. Leake, D. B., Bauer, T., Maguitman, A., & Wilson, D. C. (2000). Capture, storage, and reuse of lessons about information resources: Supporting task-based information search. To appear in (Aha & Weber, 2000).
7. Leake, D. B. Kinley, A. (1998). Integrating CBR components within a Case-Based Planner. *Papers from the 1998 Workshop on Case-Based Reasoning Integrations* (80-84). Technical Report WS-98-15.
8. Muñoz-Avila, H., Aha, D. W., Breslow, L. A., Nau, D., & Weber, R. (2000). Integrating conversational case retrieval with generative planning. To appear in *Proceedings of the Fifth European Workshop on Case-Based Reasoning*. Trento, Italy: Springer.
9. Muñoz-Avila, H., McFarlane, D., Aha, D.W., Ballas, J., Breslow, L.A., & Nau, D. (1999). Using guidelines to constrain interactive case-based HTN planning. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 288-302). Munich: Springer.
10. Reimer, U. (1998). Knowledge Integration for Building Organisational Memories. In: *Proceedings of the Eleventh Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*. Banff, Canada.
11. Sary, C., & Mackey, W. (1995). A case-based reasoning approach for the access and reuse of lessons learned. *Proceedings of the Fifth Annual International Symposium of the National Council on Systems Engineering* (pp. 249-256). St. Louis, Missouri: NCOSE.

12. Secchi, P. (Ed.) (1999). *Proceedings of Alerts and Lessons Learned: An Effective way to prevent failures and problems* (Technical Report WPP-167). Noordwijk, The Netherlands: ESTEC. [[www.estec.esa.nl/CONFANNOUN/99c06](http://www.estec.esa.nl/CONFANNOUN/99c06)]
13. Secchi, P., Ciaschi, R., & Spence, D. (1999a). The ESA alert system. In (Secchi, 1999).
14. SELLS (1999). *Proceedings of the Society for Effective Lessons Learned Sharing's Spring Meeting*. Las Vegas, NV: Unpublished. [[www.tis.eh.doe.gov/ll/sells/proceedings399.htm](http://www.tis.eh.doe.gov/ll/sells/proceedings399.htm)]
15. Vandeville, J. V. & Shaikh, M. A. (1999). A structured approximate reasoning-based approach for gathering "lessons learned" information from system development projects. *Systems Engineering*, 2(4), 242-247.
16. Weber, R., Aha, D. W., & Becerra-Fernandez, I. (2000a). To appear in the International Journal of Expert Systems Research & Applications Special issue on Artificial Intelligence and Knowledge Management.
17. Weber, R., Aha, D. W., Branting, L. K., Lucas, J. R., & Becerra-Fernandez, I. (2000b). Active case-based reasoning for lessons delivery systems. *Proceedings of the Thirteenth Annual Conference of the International Florida Artificial Intelligence Research Society* (pp. 170-174). Orlando, FL: AAAI Press.
18. Weber, R., Aha, D. W., Munoz, H., & Breslow, L. A. (2000c). An intelligent lessons learned process. To appear in *Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems*. Charlotte, NC: Springer-Verlag.

# KM-PEB: An Online Experience Base on Knowledge Management Technology

Klaus-Dieter Althoff<sup>1</sup>, Wolfgang Müller<sup>1</sup>, Markus Nick<sup>1</sup>, and Björn Snoek<sup>2</sup>

<sup>1</sup>Fraunhofer Institute for Experimental Software Engineering  
Sauerwiesen 6, D-67661 Kaiserslautern, Germany  
{althoff,mueller,nick}@iese.fhg.de

<sup>2</sup>tec:inno GmbH  
Sauerwiesen 2, D-67661 Kaiserslautern, Germany  
snoek@tec:inno.com

**Abstract:** We present a framework for assessing knowledge management (KM) technology that has been validated by using more than twenty well-known tools. Based on a commercial case-based reasoning (CBR) tool, the technology descriptions have been made available in the KM-PEB experience base over the World Wide Web. KM researchers as well as KM tool developers and/or users can search for similar descriptions and/or contribute their own ones. We think that the KM and CBR communities can benefit very much from each other, and we hope that this paper - and its underlying application KM-PEB - supports the (already started) collaboration of these communities in a constructive way.

## 1 Introduction

Knowledge Management (KM) has its roots in very different areas of research (Knowledge Engineering, Information Systems, Data Mining, Classification, Information Retrieval, Economics, Software Engineering, etc.). This causes strong complications for the comparison of different kinds of approaches. Thus, a descriptive framework is needed that is general enough to describe most of the existing approaches and sufficiently detailed to still be informative.

In this paper such a framework for categorizing and comparing KM technologies<sup>1</sup> will be described. It has been validated with more than twenty well-known tools in KM.

The framework is the foundation for an application that serves as an experience base for KM technology descriptions. This application, called Knowledge Management Product Experience Base (KM-PEB), is publicly available and, therefore, acces-

---

<sup>1</sup> By technology we mean a technique, method, or tool. A technique is a basic algorithm or a set of steps for achieving some task, a method is an organized approach based on some technique (Basili et al. 1994). A tool offers automated support for a technique and/or method.



sible on the World Wide Web<sup>2</sup>. For the implementation, Fraunhofer IESE's INTERESTS (INTElligent REtrieval and STorage System) tool was used, which is based on a commercial case-based reasoning (CBR) tool<sup>3</sup> and provides a good way to deal with continuously growing experience bases.

This is an important fact because in the initial development phase the amount of technology descriptions within the experience base is - of course - rather small, but the application is designed for being used and filled in the future.

Possible users can be anyone interested in KM. Users who provide their own developments of KM technology will benefit as much from KM-PEB as persons who need decision support for a KM tool or just want to make a state-of-the-art study. Therefore, the application is designed with an interface for adding and maintaining descriptions of KM technology.

To some degree KM-PEB continues the tradition of the CBR-PEB<sup>4</sup> application (Bartsch-Spörl et al. 1997, Althoff et al. 1999), a publicly available experience base on CBR tools and applications. The ongoing measurement and evaluation program of CBR-PEB (Nick et al. 1999) is connected to KM-PEB through a rollout. However, the underlying ontology is completely new (Snoek 1999). It has been developed based on IESE's methodology for building experience bases (Tautz 2000).

We believe that KM is not only an interesting application domain for CBR technology, but that KM methods are also helpful in CBR research and application development. In addition, CBR can be viewed (and used) as a strategy for organizational learning and KM (Althoff et al. 2000a). This creates a complicated interplay between the fields of CBR and KM, which will further increase in its complexity. One reason is the increased use of approaches like textual CBR or the combination of CBR with data mining techniques. Another reason is the already ongoing "merger" of these communities (Bomarius 1999, Schmidt & Vollrath 1999, Aha et al. 1999, Althoff & Müller 2000, Weber 2000). Therefore, we hope that this paper - and its underlying application KM-PEB - supports the collaboration of the CBR and KM communities in a constructive way.

From a CBR point of view we present an interesting application that is also of some public interest. Our experience base applications rest upon the experience factory concept (Basili et al. 1994, Bomarius et al. 1998), an approach for organizational learning from experience. This approach is refined and implemented using CBR (Althoff et al. 2000a). Already in 1997 it was argued that experience factory and CBR should (and can) be combined (Althoff & Wilke 1997, Tautz & Althoff 1997). The interesting complementarities of experience factory and CBR, which can be exploited in various ways (Althoff 1998, Althoff et al. 1998), is a very good example for the complicated interplay between KM and CBR. Common issues of KM and CBR include the acquisition, modeling, and reuse of knowledge, learning from experience, evaluation and maintenance of knowledge, as well as issues like return on investment and management of all knowledge related processes and tasks.

The paper is structured as follows. We will briefly describe the approach we took to develop our framework for assessing KM technology. We will introduce the

---

<sup>2</sup> <http://demolab.iese.fhg.de:8080/KM-PEB>

<sup>3</sup> CBR-Works from tec:inno

<sup>4</sup> <http://demolab.iese.fhg.de:8080>

framework with its subparts in detail. Then we will present the application of the framework for the analysis of 21 well-known KM tools. We also present a typical use case of the KM-PEB system that was implemented based on the framework. Finally, some conclusions are drawn.

## 2 Approach

The main steps in the KM-PEB project have been (1) the development of the descriptive framework for technology in KM and organizational learning, (2) the description of selected tools based on the framework, and (3) the technical implementation of an experience base that makes the technology descriptions available over the World Wide Web (WWW). In particular, the following tasks had to be carried out:

- Identification, analysis, and structuring of relevant literature (e.g., Goodall 1999).
- Interviews with available experts and detailed study of theoretical literature.
- Development of the basic framework structure (cf. Section 3).
- Selection of typical KM tools from different classes of KM technology.
- Information acquisition and analysis as well as description of selected KM tools.
- Elicitation of detailed information on the selected tools including the tool vendors.
- Incremental improvement of the descriptive framework using the tool descriptions.
- Implementation of an experience base that includes the tool descriptions and that enables similarity based retrieval of KM technology descriptions over the WWW.
- Going online, adding online help and an interface for the acquisition of new KM technology descriptions.
- Collecting feedback.
- A step currently being carried out: Public announcement of KM-PEB and setting-up of a measurement and evaluation program (Nick et al. 1999).

## 3 Framework

Applying the above approach resulted in a comprehensive framework for KM technology, which now will be introduced. It consists of seven main components (Fig. 1) that are adapted from Probst and Romhardt (1998). In the following we will introduce them and present the more detailed attribute level for each of the components.

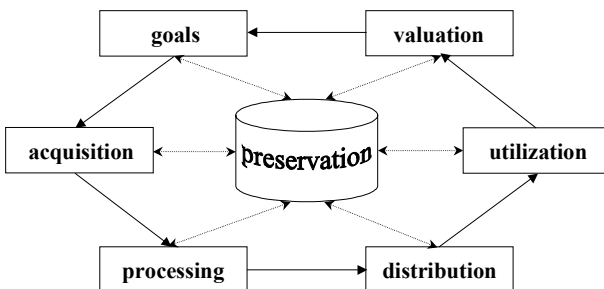


Fig. 1. The underlying knowledge management framework

## 3.1 Knowledge Management Components of the Framework

### 3.1.1 Knowledge Goals

The goals of a KM technology are subject of this framework component. The goals describe on which level what kind of ability will be built. Goals exist on different levels of abstraction, for instance, vague company guidelines or concrete task-oriented goals.

### 3.1.2 Knowledge Acquisition

This component addresses the systematic collection of knowledge. Therefore, it is first wise to identify the knowledge already existing in the enterprise, which, on the other hand, makes the existence of knowledge transparency essential. To know what the state of affairs is - relating to the available knowledge in the enterprise - knowledge maps and other similar procedures are helpful.

Because of the worldwide knowledge explosion and fragmentation, companies are not able to generate the entire know-how for the enterprise's success by themselves. Therefore, the knowledge has to be gathered using different means of procurement. Some of them will now be shortly summarized.

One idea is the acquisition of knowledge from other companies in the form of alliances and takeovers. Another way to get knowledge is the inclusion of stakeholders, a group in the environment of the enterprise with special interest in its activity. Stakeholders (e.g., customers in key positions) can be involved in the development process.

Another type of acquisition is the purchase of knowledge from external knowledge carriers like specialists. For this, a time limit on employment is possible. Out-sourcing is a further possibility to handle this kind of acquisition.

Finally, an enterprise can buy knowledge products (e.g., CD-ROM, patents, software, etc.). However, in this case one must consider that the enterprise perhaps does not have the required capabilities to effectively use these knowledge sources.

### 3.1.3 Knowledge Processing

Here the focus of interest is the generation of new abilities and products, better ideas, and efficient processes. Knowledge processing contains all management efforts that lead to the conscious production of abilities that do not yet exist internally or, further on, to the creation of abilities that do not yet exist internally or externally.

An important role is the kind of representation that the approaches will use or the tools they will support. Relationships or other associations will aid the knowledge to evolve. The question of representation is also of interest here, because for some reason it could be useful to separate the structural representation (like characterizations and relationships) from the content of knowledge.

This leads to the question of whether the knowledge structure is a crucial factor for the evolving knowledge. With the support of a useful representation a tool could, for example, analyze the knowledge structure and change it in different ways to generate new knowledge. The kind of changing reaches from adopting and aggregating relations or other associations up to restructuring the knowledge items themselves, which can also be seen as part of the whole knowledge structure.

Another way to process new knowledge is to create it explicitly. Two main classes can be regarded for an individual: Ad-hoc processing and systematic processing.

The ad-hoc processing of knowledge can be seen as an act of creation that seems to be chaotic, whereas the systematic processing of knowledge describes problem solving as a structured process in several steps.

For collective knowledge creation processing, it is also necessary to create complementary abilities in the group and to define meaningful and realistic group goals. There must be an atmosphere of confidence based on communication intensity. Furthermore, in a group it is possible (and wise) to do a post-mortem analysis after a finished project and to write it down (as lessons learned) for later use in other projects or for other groups.

### **3.1.4 Knowledge Preservation**

For later use, the storage of knowledge is a very important part of a knowledge based system. To preserve experience, information and documents require management efforts to select the knowledge that is worth preserving.

In an enterprise, every day a mass of knowledge normally arises in the form of minutes, letters, presentations given at different locations, etc. The difficulty is (see also knowledge utilization) to select between the knowledge that is worth preserving and the one that is not. The main rule for this decision is the question of whether the regarded knowledge will be used by its potential users. Otherwise, it would decrease trust in the quality of the system.

Knowledge preservation is a permanent process, which must be supported by continuous updating. Old systems are "dead systems", because omitting training leads to the unlearning of the knowledge that was developed with great effort.

It is important at this point already to restructure or repackage the knowledge. But different from knowledge processing, here it is important for the effectiveness of giving response. Also, conscious forgetting leads to a faster system and is, therefore, an important factor for knowledge base efficiency.

### **3.1.5 Knowledge Distribution**

In order to utilize isolated knowledge for the whole enterprise, distribution is essential. The main question in this case is: Who should know (or be able to do) what (and to which extent) and how should knowledge be distributed?

It is neither possible nor meaningful for everybody to know everything. The economical principle demands a useful control of the scope of knowledge distribution. Dividing knowledge into several fields is a first step in reaching this goal.

During a time when our world gets smaller through growing networks, the creation of virtual working groups is of increasing importance for knowledge distribution. Here, the people in different parts of the world are able to work together on the same projects. The creation of expert groups, in particular, benefits from this possibility, because in most cases experts are spread far apart throughout the world.

The activity in the case of publishing can be passive, that is, the people themselves have to get the knowledge if they want it. Otherwise, they get it automatically, without their own help. The informing activity is passive if the people were not informed

of new knowledge, so they have to look for new knowledge themselves. In the case of active informing, the system tells the people about the existence of new knowledge.

### **3.1.6 Knowledge Utilization**

One main purpose of the KM is placed here. The bare existence of knowledge (what the components described up to now guarantee) is not enough. Many constraints restrict the use of knowledge in the daily routine. The first problem is the dislike of people to use outside knowledge, because it is seen as an unnatural act to do so. The next borderline is the keeping of well-tried experiences, regardless of their quality, as a kind of security mechanism to avoid foreign infiltration.

In the face of these problems, the enterprise must guarantee that expensive and strategically important knowledge will be used every day. All effort is in vain if the potential users are not convinced of the system's usefulness and, thus, do not use the system.

So, one central question is whether the regarded approaches do support the motivation of people to use the offer of KM. Or, if the approaches do not provide this feature, it is interesting whether other (maybe additional) tools, which do this, can be integrated in the approach.

The next point of interest in utilization is the basic ability of choosing the right knowledge with the system. This question is close to the above-discussed representation of the knowledge. Could, for example, different knowledge items be merged together to be used as an integrated whole?

Another question depends on the ability to check out knowledge before it is used to avoid inconsistencies and conflicts with other potential users. Also, other models like transactions are possible to keep the knowledge base consistent.

### **3.1.7 Knowledge Valuation**

One of the biggest difficulties in KM is the measurement and valuation of organizational KM. Until now there is no decisive breakthrough. Unfortunately, knowledge managers cannot fall back onto a large amount of well-tried instruments as, for example, finance managers can do. Most often, knowledge and abilities cannot lead back to a measurement dimension and/or the cost of measurement would be unjustifiable high. Yet the field of knowledge valuation offers high potential.

As the knowledge goals are differentiated into strategical, tactical, and operative goals, the same is true for valuation. At the point of valuation, abstract knowledge goals like "we will become a learning enterprise" will take revenge, because there is no useful measurement for them. Only if the measurement and valuation of central aspects of KM will, in the future, become easier and gain more acceptance, will the cycle of KM be closed.

Valuation is also important for providing feedback on the knowledge base's effectiveness. Many aspects can be regarded for reaching this goal: Is it possible to analyze the system behavior? Do log files exist that generate utilization profiles? Can feedback (maybe from other subroutines logged) be used to increase efficiency by restructuring? Is it possible to qualify knowledge items using analyzable flags?

### 3.2 Framework Attributes for the Knowledge Management Components

Altogether the above-described seven KM components and the general aspects part are represented using 54 attributes with different value ranges. We will now introduce them using some kind of meta-attributes:

**Type.** This specifies the type of an attribute. Possible types are Boolean, symbol, ordered symbol, symbol-set, ordered symbol-set, text, and taxonomy.

**Possible Values.** Depends on the respective type.

**Cardinality.** Defines how many values can be chosen in parallel.

**Possibility to use “not applicable”.** In some cases, an attribute could be left empty because of its insignificance, even though it is important in other cases.

**Assessment.** Some attributes directly influence the quality of a tool. If this is so, one can describe it in more detail under this point.

**Dependencies.** Describes how a given attribute influences other attributes, if at all.

**Effects.** The other case is that a given attribute (and thus its value) influences other attributes.

**Table 1.** Overview of the framework

| Component       | No | Framework attributes                     | Type | # predefined values | Cardinality | Usage of not applicable | Assessment <sup>5</sup> | Dependencies (Attr. No.) | Effects (Attr. No.) |
|-----------------|----|------------------------------------------|------|---------------------|-------------|-------------------------|-------------------------|--------------------------|---------------------|
| General aspects | 1  | Kind of approach                         | s    | 3                   | 3           | no                      | -                       | -                        | -                   |
|                 | 2  | Developed for what goal                  | s    | 3                   | 3           | no                      | -                       | -                        | -                   |
|                 | 3  | Purpose of the system                    | t    | 13                  | *           | no                      | -                       | -                        | -                   |
|                 | 4  | Subject of knowledge                     | t    | 4                   | *           | no                      | -                       | -                        | -                   |
|                 | 5  | Main orientation                         | s    | 2                   | 1           | no                      | -                       | -                        | 17                  |
|                 | 6  | Degree of formalization                  | os   | 3                   | 3           | no                      | -                       | -                        | -                   |
|                 | 7  | Flexibility of configuration             | os   | 4                   | 1           | yes                     | left to right           | -                        | -                   |
|                 | 8  | Customizable configuration               | b    | 2                   | 1           | no                      | yes is best             | -                        | -                   |
|                 | 9  | Scalability of configuration             | t    | 4                   | 1           | yes                     | left to right           | -                        | -                   |
|                 | 10 | On which platform could it run           | t    | 24                  | *           | no                      | most                    | -                        | -                   |
|                 | 11 | Supported databases                      | t    | 20                  | *           | yes                     | most                    | -                        | -                   |
|                 | 12 | Single- or multi-user                    | os   | 3                   | 1           | no                      | left to right           | -                        | 44                  |
|                 | 13 | Supported network and transfer protocols | t    | 23                  | *           | yes                     | -                       | -                        | -                   |
|                 | 14 | Used by enterprises in practice          | te   | 0                   | *           | no                      | most                    | -                        | -                   |
|                 | 15 | Supported file formats                   | t    | 13                  | *           | no                      | most                    | -                        | -                   |
|                 | 16 | Highlights                               | te   | 0                   | *           | no                      | -                       | -                        | -                   |
| Goals           | 17 | Kinds of goals                           | os   | 3                   | *           | no                      | -                       | 5                        | -                   |
|                 | 18 | Incomplete specification                 | t    | 3                   | 1           | yes                     | left to right           | -                        | -                   |

<sup>5</sup> "Left to right" means that the rightmost value in the attribute's value range is the best value for a KM tool (a special instance is "yes is best"). "Most" means the more values are filled in the better the tool is. "-" means nothing can be stated without further context information.

|              |    |                                               |    |    |   |     |               |    |       |
|--------------|----|-----------------------------------------------|----|----|---|-----|---------------|----|-------|
| Acquisition  | 19 | Kinds of sources                              | t  | 16 | * | no  | most          | -  | 20,29 |
|              | 20 | Tolerance of incomplete data                  | b  | 2  | 1 | yes | yes is best   | 19 | -     |
|              | 21 | Acquisition methods                           | t  | 7  | * | no  | left to right | -  | -     |
|              | 22 | Connection                                    | b  | 2  | 1 | no  | yes is best   | -  | -     |
| Processing   | 23 | Knowledge representation                      | t  | 16 | * | no  | -             | -  | 26,27 |
|              | 24 | (Re)package the knowledge structure           | t  | 6  | 1 | no  | left to right | -  | -     |
|              | 25 | Evolve knowledge items                        | t  | 4  | 1 | yes | left to right | -  | -     |
|              | 26 | Coexistence of knowledge items                | t  | 4  | 1 | yes | left to right | 23 | -     |
|              | 27 | Forget unimportant knowledge items            | b  | 2  | 1 | yes | yes is best   | 23 | -     |
| Preservation | 28 | Separation of items/structure/other knowledge | t  | 7  | * | no  | left to right | -  | -     |
|              | 29 | Administrative information                    | b  | 2  | 1 | no  | yes is best   | 19 | -     |
|              | 30 | Status of knowledge items                     | t  | 4  | 1 | yes | left to right | -  | -     |
|              | 31 | Self-analyzing technical infrastructure       | t  | 6  | 1 | no  | left to right | -  | -     |
|              | 32 | Recording of new knowledge                    | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 33 | Tolerance of inconsistent knowledge           | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 34 | (Re)packaging the system structure            | t  | 5  | 1 | yes | left to right | -  | -     |
|              | 35 | Forget knowledge items                        | t  | 5  | 1 | yes | left to right | -  | -     |
| Distribution | 36 | Adressability of target groups/persons        | os | 4  | 1 | yes | left to right | -  | -     |
|              | 37 | Publishing                                    | os | 3  | 1 | yes | -             | -  | 38,39 |
|              | 38 | Retrieval mechanism                           | t  | 18 | * | yes | -             | 37 | -     |
|              | 39 | Browsing                                      | te | 0  | * | no  | most          | 37 | -     |
|              | 40 | Informing                                     | os | 3  | 1 | yes | left to right | -  | -     |
|              | 41 | Personal environment                          | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 42 | Context sensitive                             | b  | 2  | 1 | yes | yes is best   | -  | -     |
| Utilization  | 43 | Selecting/choosing possible                   | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 44 | Avoiding inconsistence                        | t  | 7  | * | yes | yes is best   | 12 | -     |
|              | 45 | Composing of knowledge items                  | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 46 | Modifying knowledge items                     | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 47 | Merging of several knowledge items            | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 48 | Modifying the system                          | b  | 2  | 1 | yes | yes is best   | -  | -     |
|              | 49 | Support for employee motivation               | b  | 2  | 1 | yes | yes is best   | -  | -     |
| Valuation    | 50 | Examine the system structure                  | t  | 4  | 1 | yes | yes is best   | -  | -     |
|              | 51 | Evaluation                                    | t  | 4  | 1 | yes | yes is best   | -  | -     |
|              | 52 | Feedback possible                             | t  | 4  | 1 | yes | yes is best   | -  | 53,54 |
|              | 53 | Analyzing feedback                            | t  | 4  | 1 | yes | yes is best   | 52 | -     |
|              | 54 | Qualifying                                    | t  | 4  | 1 | yes | yes is best   | 52 | -     |

**Legend:**  
b=boolean                      os=ordered symbol                      s=symbol  
t=taxonomy                      te=text

## 4 Applying the Framework

The application of the framework was twofold. First the framework was applied to analyze 21 KM tools. For these tools, Table 2 displays a subset of 24 of the 54 attributes presented in Table 1. The full tool descriptions can be retrieved from KM-PEB or in Snoek (1999).

Second, the framework was used for the ontology development for KM-PEB, that is, the KM components are implemented as concepts using the attributes shown in Table 1 and the values partially shown in Table 2. A simple usage scenario is given below (see also Fig. 2-4).

**Table 2.** Overview of knowledge management tools covered

|                            | General          |  | Goal setting            |  | Acquisition              |  | Processing                  |  |             |  |            | Distribution |                                                    |  |  | Utilization            |  |                             |  | Valuation                   |  |            |  |                        |  |          |  |           |  |                      |  |                             |  |                           |  |                      |  |                      |  |                          |  |            |  |                   |  |                    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|----------------------------|------------------|--|-------------------------|--|--------------------------|--|-----------------------------|--|-------------|--|------------|--------------|----------------------------------------------------|--|--|------------------------|--|-----------------------------|--|-----------------------------|--|------------|--|------------------------|--|----------|--|-----------|--|----------------------|--|-----------------------------|--|---------------------------|--|----------------------|--|----------------------|--|--------------------------|--|------------|--|-------------------|--|--------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| Knowledge management tools | Main orientation |  | Degree of formalization |  | Incomplete specification |  | Tolerance of incompleteness |  | Acquisition |  | Connection |              | Knowledge representation (Re)package the knowledge |  |  | Evolve knowledge items |  | Coexistence of knowl. items |  | Forget unimportant k. items |  | Publishing |  | # Retrieval mechanisms |  | Browsing |  | Informing |  | Personal environment |  | Selecting/choosing possible |  | Modifying knowledge items |  | Modifying the system |  | Motivational support |  | Examine system structure |  | Evaluation |  | Feedback possible |  | Analyzing feedback |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                            |                  |  |                         |  |                          |  |                             |  |             |  |            |              |                                                    |  |  |                        |  |                             |  |                             |  |            |  |                        |  |          |  |           |  |                      |  |                             |  |                           |  |                      |  |                      |  |                          |  |            |  |                   |  |                    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Legend:**

a=automatic  
ac=active  
al=all  
ap=active & passive  
f=formal  
m>manual  
n=no/none

na=not applicable  
nf=non-formal  
o=other  
pd=product-oriented  
pr=process-oriented  
ps=passive

s=structured  
sf=semi-formal  
ss=semi-structured  
us=unstructured  
y=yes  
?=undefined

Let us assume one wants to find some KM tool with specific knowledge distribution features as described in the screen shot in Fig. 2. KM-PEB then presents the ten most similar tools in its case base sorted by similarity (see Fig. 3). Simply clicking on the case number (see Fig. 3) then presents the respective tool (case) in more detail. Fig. 4 presents the knowledge distribution details of the most similar case/tool as presented in Fig. 3.



▼ **Knowledge Distribution** ?

Can the tool address groups or single persons to inform them about new knowledge or to distribute it? ?

Does the tool publish new knowledge for users/groups or has the user to look for himself? ?

What retrieval mechanism is used by the tool? ?

Is it possible to browse through the retrieved knowledge items (or through all, if no retrieval mechanism is used)? ?

Does the tool inform users/groups about new knowledge or has the user to look for himself? ?

Can the tool handle profiles for different users (or groups) to make environments for different user needs possible? ?

Is it possible to use the tool context-sensitively (for example, the leader of sales has another view to a product than a quality manager)? ?

Distribute

both

active&passive

retrieval possible  
- rule interpretation  
- relational  
- similarity based  
-- pattern matching  
-- text based

known  
none  
unknown  
undefined

yes

active&passive

yes

yes

Fig. 2. Querying KM-PEB (focus on knowledge distribution features)

| Ser. No. | Similarity | Approach Nr. | Short title                     |
|----------|------------|--------------|---------------------------------|
| 1        | 0.284286   | 13           | KA2 KnowledgeAgent              |
| 2        | 0.251429   | 7            | Intraspect Knowledge Server 2.0 |
| 3        | 0.225      | 18           | KATE Software Suite             |
| 4        | 0.214286   | 3            | Autonomy Knowledge Server       |
| 5        | 0.198571   | 5            | KnowMan                         |
| 6        | 0.194286   | 19           | EULE2                           |
| 7        | 0.182857   | 4            | grapeVINE                       |
| 8        | 0.18       | 17           | Dataware II KM Suite            |
| 9        | 0.157143   | 20           | INTERESTS                       |
| 10       | 0.154286   | 11           | RetrievalWare                   |

Fig. 3. Result for the query specified in Fig. 2

|                                                |                   |
|------------------------------------------------|-------------------|
| Addressability of target groups/persons        | persons           |
| Publishing                                     | active&passive    |
| Retrieval mechanism                            | + other           |
| Detailed information about retrieval mechanism | knowledge mapping |
| Browsing                                       | true              |
| Detailed information about browsing            | ?                 |
| Informing                                      | active&passive    |
| Personal environment                           | true              |
| Context sensitive                              | true              |

Fig. 4. Information included in the most similar KM tool of the query result presented in Fig. 3

5 Conclusion

Given the contrary goals of the different kinds of knowledge management (KM) technology<sup>6</sup>, it is obvious that even the best framework would not allow to produce something like a ranking list. With the described framework, we have tried to create a basis for describing KM technology as adequately as possible. Since we viewed the assessment as developing a technology experience base (i.e., a case-based reasoning application), we could use our methodology (Tautz 2000, Althoff et al. 1999) to come up with acceptable results (also through involving the technology providers).

Of course, there probably is some KM technology that is missing because it is perhaps on the borderline of this research field, not published yet, and/or was just overlooked. Furthermore, it is a fact that science is continuously evolving. This justifies the imperfection of the framework and requires being prepared for continuous improvement (e.g., of the value ranges) to capture the whole variety of KM and its connected approaches.

Especially the implemented application of this framework makes this expenditure a rewarding job, because the interested people can fill, use, and update this experience base with little administrative support by IESE.

We hope that this work can be a little step for KM on the long way of becoming a mature research field, and we also hope that more and more researchers, developers, and users will contribute descriptions of their KM techniques, methods, and tools. We also believe that case-based reasoning will be of increasing importance for KM both from a tool perspective as well as from a more method-oriented one.

<sup>6</sup> By technology we mean techniques, methods, and tools (see footnote in Section 1).

## References

1. Aha, D. W., Becerra-Fernandez, I., Maurer, F., and Muñoz-Avila, H. (Eds.) (1999). *Exploring Synergies of Knowledge Management and Case-Based Reasoning: Papers from the AAAI 1999 Workshop* (Technical Report WS-99-10). Menlo Park, CA: AAAI Press.
2. Althoff, K.-D. (1998). Case-Based Reasoning and Experimental Software Engineering. *Invited Talk at the 4<sup>th</sup> European Workshop on Case-Based Reasoning (EWCBR98), Dublin*, September 23-25. Available at <http://demolab.iese.fhg.de:8080/Publications/ewcbr98/>
3. Althoff, K.-D., Birk, A., Gresse von Wangenheim, C. & Tautz, C. (1998). Case-based reasoning for experimental software engineering. In M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, S. Wess. (eds.), *Case-Based Reasoning Technology*. Springer Verlag, 235-254.
4. Althoff, K.-D., Bomarius, F. & Tautz, C. (2000a). Using a Case-Based Reasoning Strategy to Build Learning Software Organizations. Accepted for the IEEE Journal on Intelligent Systems, special issue on KM and Knowledge Distribution over the Internet.
5. Althoff, K.-D. & Müller, W. (2000). *Proc. of the Second International Workshop on Learning Software Organizations*. Fraunhofer IESE, Kaiserslautern, Germany.
6. Althoff, K.-D., Nick, M. & Tautz, C. (1999). CBR-PEB: A Tool for Implementing Reuse Concepts of the Experience Factory for CBR-System Development. In Proc. 7th German Conference on Knowledge-Based Systems (XPS'99) Workshop on Case-Based Reasoning (GWCBR'99); CBR-PEB is publicly accessible via <http://demolab.iese.fhg.de:8080/>.
7. Althoff, K.-D. & Wilke, W. (1997). Potential uses of case-based reasoning in experienced based construction of software systems and business process support. In R. Bergmann & W. Wilke (eds.), Proc. 5th German Workshop on Case-Based Reasoning, LSA-97-01E, 31–38. Centre for Learning Systems and Applications, University of Kaiserslautern, March 1997.
8. Bartsch-Spörl, B., Althoff, K.-D. & Meissonnier, A. (1997). Learning From and Reasoning About Case-Based Reasoning Systems. In: P. Mertens & H. Voss (eds.), *Proc. of the 4th German Conference on Knowledge-Based Systems*, infix Verlag, Germany.
9. Basili, V. R., Caldiera, G. & Rombach, H. D. (1994). Experience Factory. In J. J. Marciniak, editor, *Encyclopedia of Software Engineering*, vol. 1, 469–476. John Wiley & Sons, 1994.
10. Bomarius, F. (1999). Proc. of the First International Workshop on Learning Software Organizations. Fraunhofer IESE, Kaiserslautern, Germany.
11. Bomarius, F., Althoff, K.-D. & Müller, W. (1998). Knowledge Management for Learning Software Organizations. *Software Process - Improvement & Practice* 4, 89-93, printed 1999.
12. Goodall, A. (ed.) (1999). Survey of Knowledge Management Tools - Part I & II, volume 8. *Intelligence in Industry*, January & February 1999

13. Nick, M., Althoff, K.-D. & Tautz, C. (1999). Facilitating the Practical Evaluation of Organizational Memories Using the Goal-Question-Metric Technique. *Proc. Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, Banff, Oct. 16-21.
14. Probst, G. & Romhardt (1998). The components of knowledge management - a practical approach (in German). URL: [www.cck.uni-kl.de/wmk/papers/public/Bausteine/](http://www.cck.uni-kl.de/wmk/papers/public/Bausteine/).
15. Schmidt, S. & Vollrath, I. (eds.) (1999). Challenges for Case-Based Reasoning - Proc. of the ICCBR'99 Workshops. LSA-Report LSA-99-03E, University of Kaiserslautern, Germany.
16. Snoek, B. (1999). *Knowledge Management and Organizational Learning*. Diploma Thesis, University of Kaiserslautern, Germany.  
<http://demolab.iese.fhg.de:8080/KM-PEB/km-ol.pdf>
17. Tautz, C. (2000). Customizing Software Engineering Experience Management Systems to Organizational Needs. PhD thesis, University of Kaiserslautern, Germany. to appear.
18. Tautz, C. & Althoff, K.-D. (1997). Using case-based reasoning for reusing software knowledge. *Proc. 2<sup>nd</sup> International Conference on Case-Based Reasoning*, Providence, RI, July 1997. Springer Verlag.
19. Weber, R. (2000). Call for Papers for AAAI-00 Workshop on Intelligent Lessons Learned Systems. <http://www.aic.nrl.navy.mil/AAAI00-ILLS-Workshop>.

# A Support System Based on CBR for the Design of Rubber Compounds in Motor Racing

Stefania Bandini and Sara Manzoni

Department of Computer Science, Systems, and Communication  
University of Milan–Bicocca, via Bicocca degli Arcimboldi, 8 - 200126 - Milan (Italy)  
tel +39 02 64487835 -- fax +39 02 64487839  
bandini@disco.unimib.it

**Abstract.** This work presents P–Race, a knowledge–based system developed for the Motorsports Department of Pirelli Tyres. The system supports the formulation of rubber compounds of tyre tread, in order to take part (and win) in motor racing. Multiple knowledge representations have been adopted and integrated into a single Case–Based Reasoning (CBR) computational framework in order to capture the different competence involved in the decision making process. Moreover, a dedicated representation formalism called Abstract Compounds Machine (ACM) has been introduced in order to represent, compute, and integrate in the CBR architecture the core knowledge regarding rubber compounds. The result is a general case–based architecture where the adaptation step is demanded to a component based on the ACM model.

## 1 Introduction

In motor racing, the role of tyres is crucial for the results of the competitions. Among the parts that build a tyre, *tread* is one of the most important. The basic material composing tread is called in jargon *batch*. Batch is a mixture of different elements, that determine its major properties, described by a *recipe*.

A set of standard lab tests is usually performed in order to obtain the best possible performance from a tread. However, most of the tests cannot be applied to tread dedicated to racing, because of its chemical characteristics and the raw materials used. Therefore, the performance of tread can be assessed only in trials or directly during the race. The evaluation of the performances depends on several factors. The most important, changing with each single race, are car set–up, road ground, geometrical profile and severity of the tracks, weather conditions, and skills of the racing team. Tyres for motor racing are designed by a team composed by different people (race engineers, tyre designers and compound designers), which act according to their experience on the field and their knowledge about a very complex decision making problem. More in detail, the choice of the tread batch for a competition falls on the race engineer and the compound designer. Their decision is usually strongly dependent on performance and results obtained in previous races on tracks they consider similar to the current one. Moreover, even if in a previous race their choice had led to success, the

improvement of some factors (grip, warm-up, thermal and mechanical stability, resistance to wear) is anyway advisable, because of possible improvements in the products of other tyre makers. Therefore, the choice of a tread for a given race depends mostly on the results of previously solved cases: the general problem solving mechanism used by race engineers and compound designers is strongly based on reasoning about past cases in order to solve a new case. Also, the use of *episodic knowledge* is one of the main characteristics determining the choice of the tread batch for motor racing.

The P-Race system is a Case-Based Reasoning system dedicated to design, that supports the activity of the race engineer and the compound designer. It has been integrated in the general information technology environment of the Motorsports Department of Pirelli Tyres.

In the literature, other CBR-based systems devoted to chemical formulation are presented in [1], and [2] (where an adapter module is explicitly shown).

The main purpose of this paper is to describe the features of the P-Race system. The design activity for tread batch will be outlined in the next section. In section 3 the general architecture and the use of P-Race system will be described. Finally, some concluding remarks will be given, including an outline of the benefits deriving from the application of the system.

## 2 Design Activity of Tread Batches

A large number of teams is involved in many motor sport championships. A key factor for the results of races is the quality of the tyres, that are supplied by many different producers. Among these, one of the most important in championships such as Sports Racing World Cup (SRWC) and American Le Mans Series (ALMS) is Pirelli Tyres. Pirelli Tyres includes a division entirely dedicated to racing, composed by race engineers and compound designers that, before each race, decide what kind of tyre has to be supplied to the racing teams. The choice could be the *reuse* of a kind of tyre already used in past races, the *adaptation* of a tyre for the current race, or the *creation* of a brand new tyre.

The design of a new tyre usually involves all the components (batch, structure, design, and so on), while the adaptation of a tyre usually changes only the chemical formulation of batch. In the latter case, the compound designer adapts the recipe according to the change in performance needed by the race engineer. The decision of designing a brand new tyre is usually taken only when the requests of the race engineer cannot be fulfilled by the compound designer.

The race engineer decides whether to reuse a tyre or modify it, starting from his own experience in racing and the evaluation of past performances. In particular, he considers past cases that he deems similar to the current one according to different factors:

- the profile of the circuit and the road ground;
- telemetric data collected in past editions of the same race, and the aerodynamic characteristics of the cars;

- the performance of tyres supplied by other makers;
- pilots’ comments about the performance of the tyres;
- results of past races;
- weather conditions.

At this point, according to the principle “similar solutions for similar problems”, he makes his decision. If it means modifying an existing batch, he seeks the help of the compound designer. The latter acts according to his own knowledge about the chemical properties of the materials and the relation between tyres performance and chemical formulation.

### 3 General Architecture of the P–Race System

The design activity of tread batches for tyres dedicated to motor racing at the Motorsports Department of Pirelli Tyres can be represented in terms of Case-Based Reasoning in Design (CBRD) [3,4]. As in other cases of design activity supported by CBRD, the P–Race system supports the Motorsports Department team both for the reuse of a solution adopted in past experiences and the generation of new solutions adapting an old case to the current problem. In particular, P–Race assists in their activity the two main characters of the team: the race engineer and the compound designer.

The general architecture of P–Race is outlined in Figure 1. It can be divided into three main conceptual parts (drawn by dotted lines in the figure). Part A contains the main components devoted to the race engineer: a dedicated user Interface designed in order to access both the Tracks Description component and the Races Data component. Data contained in the last two components are indexed in order to be transformed into cases.

Part B consists of the main components supporting the activity of the compounds designer: a dedicated user Interface, the ACM component which represents the adapter and comprises the generator of recipes and an explanation module, and the integration interface for access to the Recipes Archive belonging to the information system of the company.

Part C is the Case-Based Reasoning core, composed by a Case Memory component and the Indexing, Reuse, Retrieve, and ACM Adapter modules.

#### 3.1 P–Race Support to the Race Engineer

The race engineer work is supported by a dedicated module of the P–Race system. The user can interact with the system in three different ways, that is, he can input a track description, activate the case-based engine, or update the database. A Knowledge Acquisition (KA) module has been designed in order to allow users to express their knowledge about tracks both in qualitative and quantitative ways, avoiding as much as possible subjective descriptions. The KA module allows a representation of tracks decomposed in *blocks*. Each block is characterized by a type (stretch, bend, chicane, and so on), and by the level of

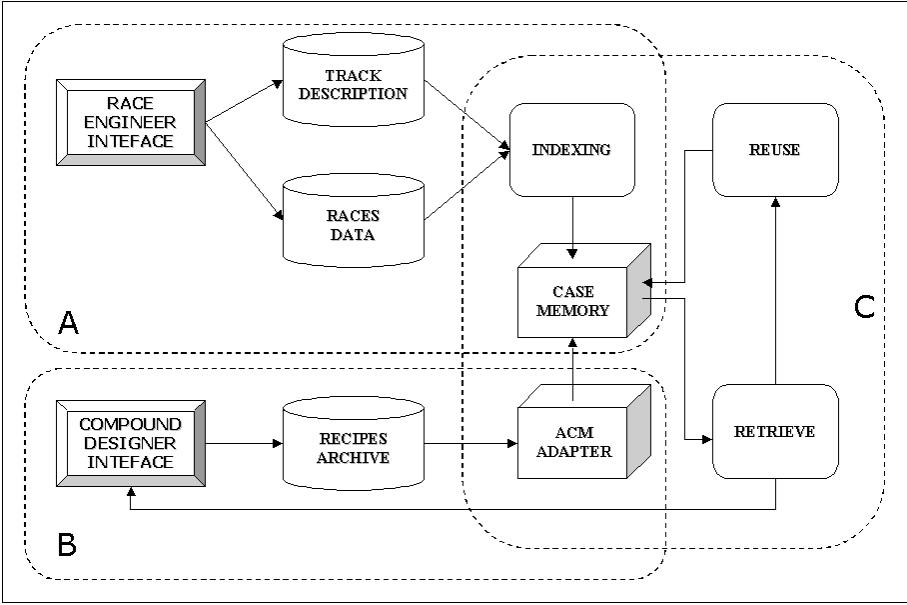


Fig. 1. The architecture of P-Race

stress tyres have to withstand (determined also by the road ground), expressed with numerical quantities. The result is a representation of the tracks that captures the experience and the knowledge of the race engineer in terms of crucial information about the track (e.g., the features of track bends in terms of their severity and the required tyre performance, the characteristics of the track surface, the thermal variation from a straight stretch to a bend and vice versa). Moreover, the representation allows the user to compare tracks according to their profiles. Figure 2 shows the interface dedicated to the race engineer.

The update of the database containing all meaningful data about the racing activity (dates, kind of championship, car, team, drivers, trial times, warm-up times, race times, coded recipe of the used tread batch, coded information about the tyre structure, tyre performances, and so on) is usually done directly on the field, that is, at the circuit during the competition.

The purpose of the activities described above is to provide the system with all the information needed by the user for the choice of the most suitable tyre for the race.

The reasoning process starts with the representation of the current problem (given by the user) as a case to be solved. A case represents a set of chronometrical measurements, concerning a race or a trial, relevant for the performance or the technical solution adopted. As in any CBR system, the three major parts of a case are problem/situation description, solution, and outcome [5]. In P-Race



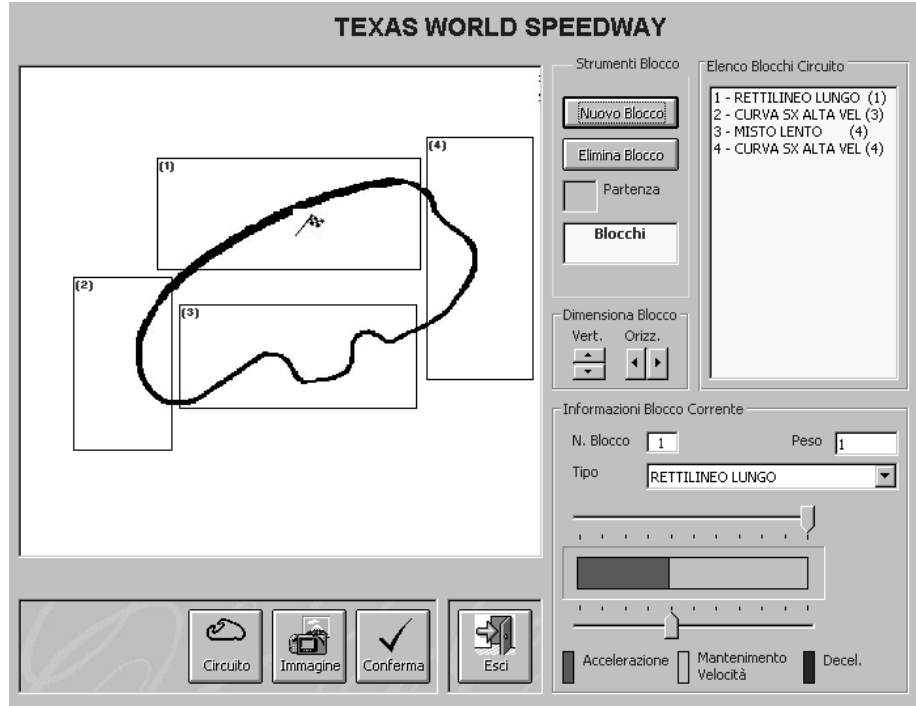


Fig. 2. User interface for the track description

the description of the problem contains descriptive information (date, time and location of the event) and a set of parameters used by the system to retrieve from the case memory the most similar cases. The solution for a case describes the coded recipe of the batch for that case, while the outcome represents the resulting state in terms of performances obtained when the solution was applied. Starting from the new description, the system examines the case base (which indexes and structures in cases the pragmatic features of a race to be used in the design of solutions for a new race) containing past cases already solved, and proposes a list of solutions (the most similar cases). In the P-Race system, the similarity function has been defined as the weighted sum of differences between attributes. Case retrieval is based on knowledge about tracks, weather conditions, and type of track surface. The list of solutions proposed by the system could, at this point, include a feasible solution for the problem at hand that could be directly applied. Otherwise, an adaptation process has to modify one of the proposed solutions.

### 3.2 P–Race Support to the Compound Designer

A dedicated module of the P–Race system has been developed in order to support the compound designer in the chemical formulation of a batch, and to allow him the access to the information contained in a database of recipes and other confidential data about raw materials. The formulation of a new compound is guided by the requests of the race engineer, that asks for the improvement of some performance of an existing batch. Starting from the solution and the outcome of a case retrieved by the P–Race module dedicated to race engineer, the compound designer examines the recipe and the race conditions in order to obtain a solution for the problem.

The decision process of the compound designer can be divided in three separate stages:

1. *Batch evaluation*: the expert is usually able to assess the performance of a compound from its parameters. The evaluation of the compound designer is usually different from the one given by the race engineer: the former judges according to his theoretical knowledge about materials, while the latter examines the results of the races.
2. *Definition of the objective*: according to the results of the analysis of the previous point, and to the information about the race context, the expert decides which property of the batch has to be changed in order to obtain the desired performance. The properties usually involved are grip, thermal stability, mechanical stability, and warm-up. At the end of this stage, the compound designer has a set of possible options leading to the needed improvement of the performance.
3. *Choice of the ingredient*: finally, the compound designer describes the batch recipes contained in the archive of the company as lists of ingredient quantities. Then, according to the chemical and physical properties of the raw materials, chooses an ingredient, and decides whether (and how) it has to be changed in quantity or if it must be substituted by another one.

P–Race supports the activities described above with an adapter module. The *ACM Adapter*, activates the integration interface with the recipes archive of the company in order to provide to the ACM component the decoded recipe expressed in terms of quantity of ingredients (see *Description Rules* in the next subsection). That is, it modifies the recipe of the proposed batch in order to improve the performance observed in the outcomes of the past case, or to obtain new performances in relation with the description of the new case.

### 3.3 The ACM Adapter Component

The Abstract Compound Machine (ACM) [6] is a model created for the representation and the computation of the knowledge involved in the chemical formulation of a compound. It is derived from the CHAM (Chemical Abstract Machine) formalism [7]. The CHAM is an abstract model that introduces a chemical metaphor in order to deal with rewriting techniques in concurrent systems. The

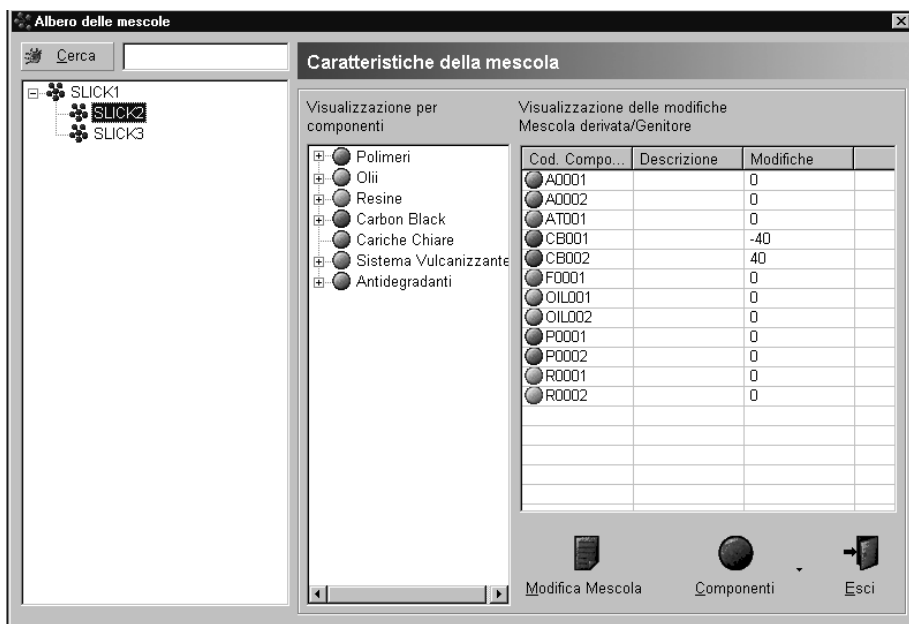


**Fig. 3.** Presentation of the adapted recipe (all data in this figure are fictitious)

general computational model offered by the CHAM allows reaction systems to be described. In our case the reaction system is represented in a symbolic model and its semantic corresponds to a real case of chemical reaction. Moreover, reactions in the ACM correspond to a rewriting rule based model. In particular, the implementation of the ACM model presented in this paper regards the formulation of rubber compounds. Since recipes are strictly confidential, only an abstract description of this mechanism and some simple examples can be given. In the ACM model, a recipe of  $n$  ingredients is a finite non ordered set  $\{Q_1 \dots Q_n\}$ , where each  $Q_i$  represents the quantity of the  $i$ -th ingredient. A given ingredient belongs to one or more families of ingredients. Each family  $F_k$  is described by a set  $\{P_1^k \dots P_m^k\}$  of attributes. Therefore, each element of a family (an ingredient) is described by a value  $V_{ij}^k$  for each of its own attributes  $P_j^k$ . The values of each attribute belong to the interval  $[V_{ij}^k - T_j^k, V_{ij}^k + T_j^k]$ , where  $T_j^k$  is a constant of tolerance. If an ingredient  $i$  does not belong to the family  $F_k$ , the corresponding values  $V_{ij}^k$  are undefined. Starting from a recipe  $R$ , a modified recipe is a recipe  $R'$  where some quantities have been changed. Modifications follow the application of four sets of rules, where rules take the form of productions:

1. *Description Rules*, describing a recipe taken from the company database expressed in terms of quantity of ingredients, according to the ACM model.

2. *Performance-Properties Rules*, defining which changes are needed in the chemical-physical properties of a batch defined by a recipe, in order to obtain a change in performance (for example, 'in order to increase thermal stability, hysteresis must be decreased').
3. *Ingredients-Properties Rules*, defining which attributes of the ingredients of a recipe are involved in the variation of the chemical-physical properties of the batch (e.g., 'in order to increase hysteresis, transition glass of polymer is involved').
4. *Formulation Rules*, generating a modified recipe  $R'$  starting from  $R$ . Three types of formulation rules have been defined:
  - (a) *Substitution*, replacing the quantity of an ingredient  $i$  with an equal quantity of another ingredient  $l$  of the same family  $F_k$  (chosen by the Ingredients-Properties Rules), in order to change the value of one or more attributes ( $V_{ij}^k$ ):

*if* ( $Q_i \neq 0$ ); ( $i \in F_k$ ); ( $l \in F_k$ ); ( $|V_{ij}^k - V_{lj}^k| > T_j^k$ ) *then*

$$\{Q_1, Q_2, \dots, Q_{i-1}, Q_i, Q_{i+1}, \dots, Q_l, \dots, Q_n\} \rightarrow \\ \{Q_1, Q_2, \dots, Q_{i-1}, 0, Q_{i+1}, \dots, Q_i + Q_l, \dots, Q_n\}$$

- (b) *Increase in quantity*, adding to the quantity of an ingredient a given constant  $U_k$ , defined according to the family  $F_k$  of the ingredient:

*if* ( $i \in F_k$ ) *then*

$$\{Q_1, Q_2, \dots, Q_{i-1}, Q_i, Q_{i+1}, \dots, Q_n\} \rightarrow \\ \{Q_1, Q_2, \dots, Q_{i-1}, Q_i + U_k, Q_{i+1}, \dots, Q_n\}$$

- (c) *Reduction in quantity*, decreasing the quantity of an ingredient by a constant  $U_k$ , defined as in the previous point:

*if* ( $i \in F_k$ ); ( $Q_i > U_k$ ) *then*

$$\{Q_1, Q_2, \dots, Q_{i-1}, Q_i, Q_{i+1}, \dots, Q_n\} \rightarrow \\ \{Q_1, Q_2, \dots, Q_{i-1}, Q_i - U_k, Q_{i+1}, \dots, Q_n\}$$

The implementation of the module described above has been developed in a Java production rules environment (JESS). The knowledge base has been partitioned in *knowledge sources* corresponding to the defined type of the rules formalized above. The firing of these knowledge sources starts from the application of the *description rules*, that split the coded recipe contained in the retrieved case into the quantities of its ingredients, invoking the integration interface to the Recipes Archive component. Then, the *Performance-Properties Rules* knowledge source is activated, in order to determine the needed chemical-physical properties of the batch starting from the required performance. From the global properties of the batch, the *Ingredients Properties Rules* knowledge source finds out which ingredients are involved in order to obtain a variation of

the chemical–physical properties satisfying the required performance. The *Formulation Rules* knowledge source is the core of the ACM Adapter component. It modifies the recipe applying *Substitution*, *Increase in quantity* or *Decrease in quantity* rules. The choice of the rules determines how the recipe is adapted to the current case. Figure 3 shows a snapshot of the compound designer interface illustrating a modified recipe.

## 4 Conclusions

The P–Race Project is the result of the cooperation between the Laboratory of Knowledge Engineering at Department of Computer Science, Systems and Communication of the University of Milano–Bicocca and the Motorsports Department of Pirelli Tyres, where it is currently in use.

The Motorsports Department includes people entirely dedicated to the racing activity (race engineers, tyre engineers, and other technicians) and people shared with departments dedicated to large scale production, like rubber compound designers.

Since racing usually involves tests of compounds under extreme stress conditions and of new solutions in compound design, validation of properties of raw materials, and evaluation of the declarations of the ingredients suppliers and of new products, large scale production can derive some significant benefits. Several large scale products have been designed and produced taking into account tests, results and suggestions derived from the car racing world. The most meaningful benefits obtained from the development and the use of P–Race can be summed up as follows:

- the organization of information and data concerning the racing activity in a conceptual framework, corresponding to the main characteristics of the episodic knowledge involved (Case–Based Reasoning - CBR);
- the development of similarity criteria among circuits and information about races in order to optimize the reuse modality;
- the formalization of a knowledge model for the chemical formulation of rubber compounds (Abstract Compound Machine - ACM);
- a support for the decision making process proposing solutions, i.e. adaptation changes in a recipe in order to satisfy some required performance;
- the creation a computational framework shared by all the members of the Motorsports Division of Pirelli Tyres.

The future development of the system will include the integration of P–Race with existing software systems devoted to the acquisition and the description of track data with telemetric devices. This integration will permit to support qualitative representations (fuzzy descriptions) of knowledge concerning tracks described by quantitative data.

## References

1. W. Cheetham, J. Graf. 1997. Case-Based Reasoning in Color Matching. In D. B. Leake, E. Plaza (Eds.), *Case-Based Reasoning Research and Development*, Proceedings of the 2nd International Conference on Case-Based Reasoning, Springer-Verlag, Berlin. 349
2. S. Craw, N. Wiratunga, R. Rowe. 1998. Case-Based Design for Tablet Formulation. Proceedings of the 4th European Workshop on Case-Based Reasoning, Springer-Verlag, Berlin. 349
3. M. L. Maher, M. Balachandran, D. M. Zhang. 1995. Case-Based Reasoning in Design. Laurence Erlbaum Ass. Pu., Hove (UK). 350
4. K. Börner. 1998. CBR for Design. In M. Lenz, B. Bartsch-Spörl, H. Burkhard, S. Wess (Eds.), *Case-Based Reasoning Technology*, Lecture Notes in Artificial Intelligence 1400, Springer. 350
5. J. Kolodner. 1993. *Case-Based Reasoning*. Morgan Kaufmann Pu., San Mateo (CA). 351
6. S. Bandini, S. Manzoni. 1999. Modulo di Adattamento e Modifica Ricette, Pirelli P-Race Project. Internal Report (FR2). 353
7. G. Berry, G. Boudol. 1992. *The Chemical Abstract Machine*. TCS 96, Elsevier. 353

# Supporting Tourism Culture via CBR

Enrico Blanzieri<sup>1</sup> and Alessandro Ebranati<sup>2</sup>

<sup>1</sup> ITC-IRST

via Sommarive 18, I-38050 Povo, Italy

`blanzier@irst.itc.it`

<sup>2</sup> ITC-IRST and Università di Trento

via Inama 5, I-38100 Trento, Italy

`imbra@irst.itc.it`

**Abstract.** We present COOL-TOUR, a WEB-based Case Based Reasoning system for tourism culture support. Its main goal is to build the community of tourists that are interested in a geographic area. Building communities is relevant for e-commerce and CBR can support it effectively. Users interact with the system entering tours for accessing auxiliary services. The tours are stored in a case base and other users can retrieve and adapt them. The community is built around these stored experiences. The implemented version of COOL-TOUR deals with Mountain Bike tours, providing a sketch-based interaction interface. The spatial data similarity assessment is based on spatial indexing.

## 1 Introduction

The complex relationship between Information Technology (IT) and Tourism has been widely evaluated and described in a recent book by Hannes Werthner [17]. The author emphasizes the huge economical potential of the application of off-the-shelf or state-of-the-art IT to the tourism sector. In this scenario, technologies like telecommunications or Artificial Intelligence can play important roles in the development of innovative systems for the tourism industry. From an AI perspective, intelligent or adaptive interfaces, the use of data mining techniques and the application of the agent paradigm are among the most popular approaches. Less attention has been devoted to the application of Case Based Reasoning (CBR). In fact, the potential of CBR technology in the tourism domain, and more generally in e-commerce, has still to be exploited. Important exceptions are FIND-ME systems and BUTLER agents for supporting car and restaurant choices respectively [8], a system for CBR retrieving of last minute travel offers [18] (<http://www.reiseboerse.com/>) and a CBR user profiling architecture [16]. From the e-commerce point of view a successful CBR application has been developed for renting flats in Dublin (<http://www.hookemacdonald.ie/>) and recently an approach that combines collaborative filtering and CBR for expressing recommendations on television programs has been proposed [15]. For sake of completeness it is worth to say that CBR has been widely exploited in call-centers for diagnosis and help on the use of sold goods, i.e. in the post-sale phase of trading (see for instance [19]).

This paper describes the architecture of COOL-TOUR, a system aimed at exploiting a CBR approach for an e-commerce system. The basic idea is to support the development of the virtual community of the consumers of goods or services. The approach is well suited in the case of goods or services with a complex or preference-driven use, e.g. goods needing some form of configuration or aggregation. That's the case of tourism goods where the aggregation of basic products in complex products (a travel package, an integrated destination or a tour) is the common way of producing. CBR technology can be exploited to present the results of previous aggregations done by other users or by professionals.

In order to test the basic ideas and the architecture, a prototype system on Mountain Bike Tours that supports the culture of bikers has been developed. Relevant characteristics of the system are the sketch-based interaction and the similarity assessment on spatial data. The system will be included in a site for tourism promotion of Provincia Autonoma di Trento, Italy.

The paper is organized as follows: the next section introduces the main features of a system aimed to support tourism culture, Section 3 presents the architecture and, finally, Section 4 illustrates the details of the implemented system.

## 2 A System for Tourism Culture Support

The main feature of the system is to realize an IT support for the culture of the tourism community that uses tourism services in a particular geographic area.

The basic idea is to facilitate the natural tendency of the tourist to tell other tourists about their vacations or vacation projects. Usually tourists exchange information about their vacations giving each others hints and suggestions. Friends and relatives are a common source of trusted information. An IT support can expand this interaction also to a wider group. The Internet is the natural medium for such a system, for it provides low-cost access on a world-wide scale.

From the perspective of e-commerce, community building is relevant. Usually e-commerce is composed of three phases [18]: pre-sale, sale and post-sale. In the pre-sale phase the emphasis is on the gathering of information about the product. The sale phase is devoted to the negotiation and securing of transactions, while the goal of the third phase is to assist the customer during the use or the consumption of the product. It has been suggested that the building of a virtual community should be a fourth step of an e-commerce system [14]. The advantages of building a community around a specific product or class of products are that they induce trust and fidelity keeping low the level of uncertainty that characterizes the e-commerce transactions. The system covers only the pre-sale phase of the e-commerce and the phase of community building. In general, the post-sale part proves to be usually very thin in the tourism domain because the tourist cannot even access the net during the travel. Finally, in our approach the sale part is absent because we assume that the final provider is interested in a portal site to the tourism services of the particular area that integrates or provides links to other commercial services. This is particularly true in the case



of an institutional final provider that does not sell tourism goods but promotes the activities of other actors.

Personalization is another important aspect of an e-commerce system. Quite often user profiling techniques are exploited in e-commerce sites (see for example [2]). Collaborative filtering for recommendations is also a popular technique [3]. The system does not choose an aggressive and pro-active personalization strategy but permits the user to explicitly store and edit user's data on travel experiences. Different to the other approaches where the personalization process is invisible to the end user, the user does know that the information is stored and will be used by others.

The strategical goals of the system can be summarized in four main points:

- *Provide Aggregation and Information Services for Tourists and Professionals.* One of the main characteristics of the tourism goods is the aggregation of initially unrelated goods such as accommodations, tickets for travel or shows, into an aggregate service. The system should help the tourist to aggregate goods giving also information or links to information and services related to single products. In this way the system supports the pre-sale phase (first e-commerce phase) of the tourism products and provide the linkage to the sale and post-sale phases (second and third e-commerce phase). Finally, the system provides the professionals with information about the tendencies and preferences of the potential customers in order to help the professionals to provide up-to-date services.
- *Building the Culture of the Tourists.* The aggregations that the tourists enter allows a new user to exploit the normally-distributed culture of the community of people who spend their vacations in the same area. The system should exchange services with information in order to motivate the tourists to enter their own experiences. The services provided must be valuable enough to justify the entry effort done by the user. Community building represents the fourth phase of e-commerce.
- *Gathering Subjective Information.* The system helps institutional providers to pursue their institutional goals and also to gather data on the demand in order to explore the market for new goods and services. In particular, the aim of our system is to bridge the gap between objective and subjective data. The former are the data that do not depend on the single user (the availability of a room or the presence of an exhibition or a cultural event) whereas the latter are the data that depend on the preferences of the single tourist. For instance, the information about a room price in a particular hotel is objective, while the sequences of hotels a tourist chose in her particular tour in a valley is subjective
- *Provide a Portal Site.* In any case the system is a portal site for accessing the tourism services and sites of the area, providing the linkage to available objective data on tourism services. For instance: data integration of *Geographic Information Systems* for the tourism activities that refers to a particular area; *Reservations Systems*; *Meteorological data* like weather forecasts, or generic information like events, history, culture. For an institutional

provider the system will support its activities of promotion and intermediation. In particular, the system should also improve the trust of the tourist on the information provided by the site and on the quality of the tourism services provided by the whole area.

It is clear that the schema so far described illustrates a very general framework and different instantiation could be used in a wide range of domains, i.e. any culture scenario.

### 3 The COOL-TOUR Architecture

There are many ways of supporting a virtual community and its culture on the Internet. We can classify them depending on the level of explicit communication among the participants. When the explicit communication is the main feature (like in the case of mailing-lists, newsgroups, forums, chat-lines) the system simply provides a medium for the communication between the users. Most of the attention on virtual communities on the WEB is focused on this kind of approach. At the other end of the scale, there is the case where communication is implicit and the interaction happens mainly via the reuse of the data provided by other users. Collaborative filtering is an example of the latter case (like the recommendations in the site of amazon.com or the bookmarks filtering in hotbot.com).

COOL-TOUR supports the culture without too much interaction between the users by means of data explicitly collected from the users themselves. Case Based Reasoning [1] appears to be a natural choice. CBR is a technique that derives its effectiveness from the idea that similar problems have similar solutions. The basic CBR cycle requires the phases of retrieving, reuse, revise and retain. In these phases respectively a similar case is retrieved from the case base, is reused and is revised to the problem at hand and possibly retained in the case base. In our system, the CBR engine allows for the basic functionalities of storing travel experiences and travel projects. The different experiences, namely tours, stored in the case base form the core of the system. The CBR interaction on experiences explicitly entered by the users is the way in which the system supports the culture of the community.

The CBR application needs to be integrated with other technologies in order to solve the problems of a real-world application.

- *Client-Server WEB application.* We use a three-tier architecture for supporting all the functionalities. The data reside on a Data Base Management System, the business logic on an application server and the presentation part is supported by the client. The advantage is the separation among the modules. A middleware layer in the back-end is useful for integrating the application modules to external servers.
- *Human-Computer Interaction.* The interaction between the user and the system is critical. Even without introducing the notion of persuasive technology [5] a system that has among its goals to attract potential tourists

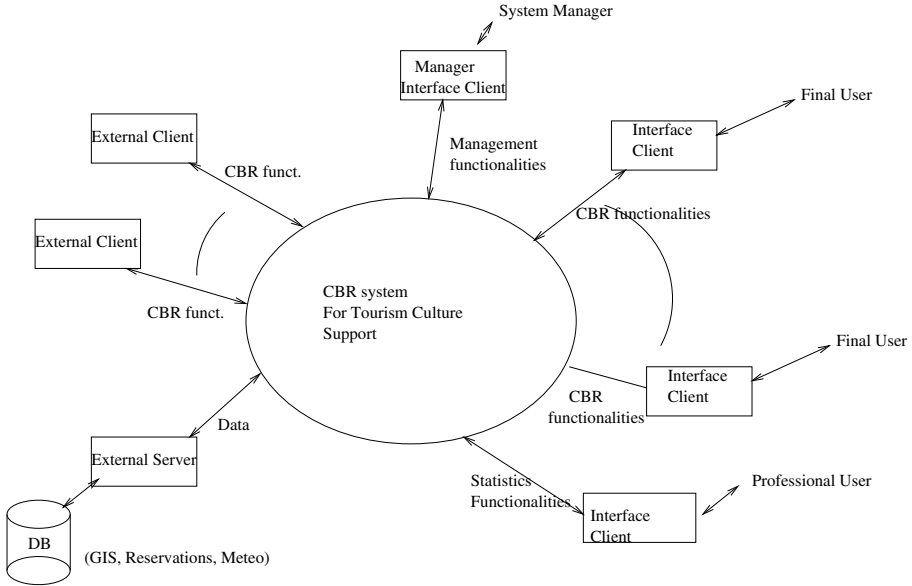


**Fig. 1.** External Architecture. The system in a client-server perspective

has to deal with Human-Computer Interaction problems and usability. The CBR paradigm provides also a model of interaction, however good interface and graphic layout are necessary.

- *Data Integration.* The need to access data from different kind of sources leads to the problem of dealing with heterogeneous data. The work done in data integration [7] is useful in order to solve the problem of interconnection and multiple interpretation of data.
- *Geographic Information Systems (GIS).* From the tourism point of view geographic data are relevant so GIS data and a connection with a GIS engine are integrated in the system architecture.

The system has interactions with different kind of users (user, professional user, system manager) as well as it is connected with other external systems playing the roles of servers or clients. Fig. 1 shows the external architecture.

In a three-tier architecture the logical architecture is mapped as shown in Fig. 2. A middleware layer provides the necessary connections to other servers.

The logical internal architecture of the system is organized in a set of three interface components and four main modules. The interfaces are:

- *Server Interface.* It connects the system with an external server.
- *Client Interface.* It provides an external client with the CBR functionalities (retrieve by similarity, adaptation) and connections with external services.
- *User Interface.* The User Interface module provides different interaction depending on the type of user and in general it provides registration and identi-

fication utilities. For the normal user the registration is optional. The module permits entry of a partially specified case as well as the transfer to the Case Editor. It also allows to trigger retrieval and adaptation and the access to auxiliary services. The system manager is qualified to maintain the case base and the user data.

The four main components are a Case Editor, a CBR module, a User Data Manager and an Auxiliary Services component.

- *Case Editor*. The Case Editor allows the user to enter, store, update, delete and copy cases and it also stores and delete partially specified cases (PSC) entered for query purposes.
- *CBR Module*. It uses a case base and three submodules for retrieval, adaptation and case base maintenance. The *Case Base* contains experiences in form of cases. There are two different collections: private and public cases. The *Retrieve* submodule receives a partially specified case and returns similar cases extracted from the collection of public cases and from the collection of the specific user. The *Adaptation* submodule receives in input a partially specified case and a similar case and returns an adapted case. The *Case Base Maintenance* submodule allows for the validation and the passage of private cases to the public case base and vice versa.
- *User Data Manager*. This module allows the management of the *User Data* such as identifiers, passwords and grants on different data. The *Register* and *Identify* modules permit registration and identification of the user.
- *Auxiliary Services*. It is the part of the system that provides the additional services. The *Service Module* is the part of the system that performs the additional service that motivates the users to entry their own experiences. The *Web Server* represents the part of the system that manages the web pages.

## 4 The Implemented System: Mountain Bikes Tours

This section presents a brief outline of the chosen prototype: a system for supporting choice and scheduling of Mountain Bike Tours via a CBR interaction. The choice of the bike tours domain is motivated by different and complementary reasons. First comes the centrality of the notion of tour in the tourism domain, then follows the criticality of the goal of supporting the culture of a virtual community and finally the local interest in mountains and biking.

The main hypothesis that underlies our approach is that the basic notion of the tourism experience is the tour itself. By tour we mean here a sequence of stretches on some road graph in a particular geographic area. From the tour it is possible to point to a lot of different and relevant tourism information like food and accommodation. We chose the bicycle tours because they are significant by themselves, even using only geographical information. In this way we can test the relevance and effectiveness of this representation of the tourism experience

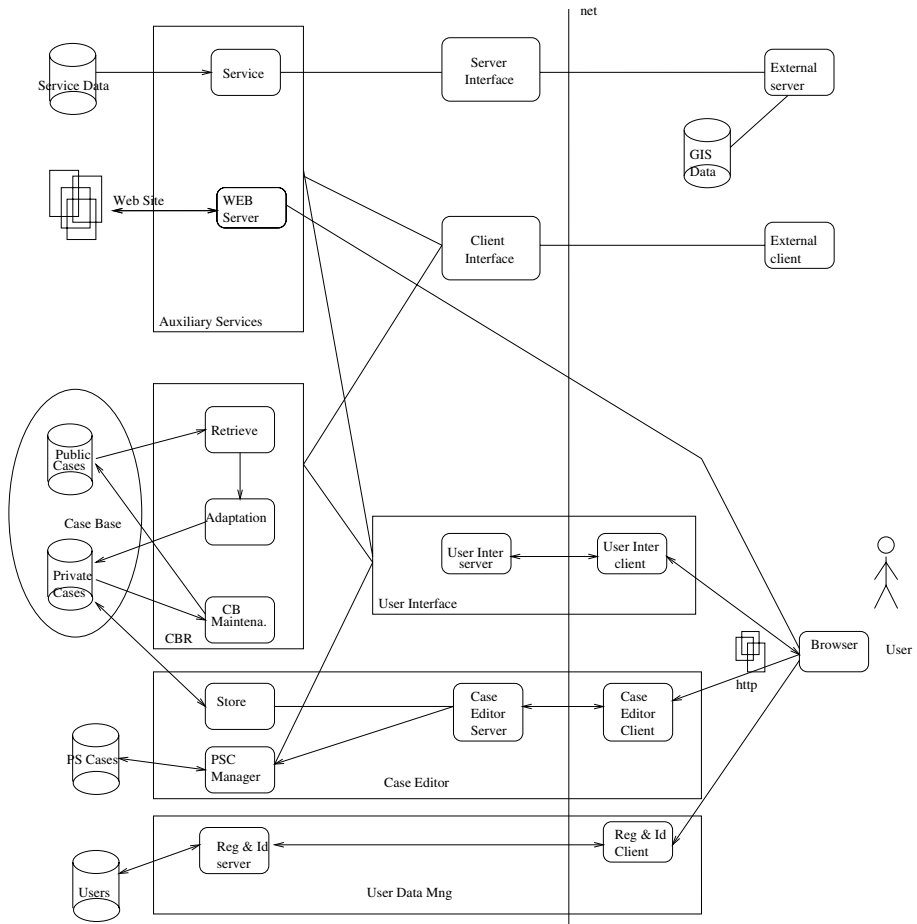


**Fig. 2.** COOL-TOUR Architecture

without needing to add too much auxiliary information or support a real e-commerce sale or reservation phase. It is worth noting that the concept of tour on a geographical map covers a lot of different tourism experiences like the visit to the monuments of a city, a car travel connecting different destinations or an hiking excursion.

The highest priority of the general system is to build a virtual community and support its culture. From our point of view the community of bikers has some characteristics that simplify the task. There is a diffuse biking culture supported by magazines and books. Bikers are somehow a subculture already with shared knowledge and a common jargon. They are also willing to tell each other about their biking experience. Some of them are interested in technology like high tech bikes or cardiac frequency monitors.

The development of the system was funded by CARITRO Foundation, a local institutional agency of Trento, so the target of the system is the local community. The mountain scenario is central in the tourism field of the local area. Moreover there is a strong interest and attention towards promoting and pushing biking, knowing that it can play a major role among the tourism activities during the summer.

The CBR approach affects the basic interaction of the user with the system. The user can enter a partially specified case (PSC) by drawing a tour on the map. The PSC can be completed by the user and stored as it is among her personal experiences or can be used to retrieve similar tours entered by other users. The system also supports the editing, namely the adaptation, of the retrieved cases. As an auxiliary service the tourist can get the altimetric profile of the tour. Altimetry is so important for bikers in a mountain area that it justifies the entry effort.

The users are registered with a login and a password in the case they want to edit their own tours. The identification of the users allows the biker to manage his own private cases; a case can be edited and published, i.e. explicitly marked by the user so that it can be shared with the community. The ownership of the cases is respected and the system can offer the user a personalized view on the case base (the user sees her own cases + the public cases).

In order to be user-friendly, the editing module exploits a sketch-based interaction. Spatial-Query-by-Sketch [4] was introduced in order to improve the communication between the user and the GIS system. Fig. 3 shows snapshots of the user interface.

The user who wants to describe a tour interacts with an applet downloaded in her web-browser. By issuing a basic set of commands (pan, zoom, unzoom, full-width) she navigates a map representing the local territory with roads, cities and other relevant geographic information. She can draw a path by dragging the mouse over the map, obtaining a “scribble”. The client-side of the user interface module has GIS capabilities. The coordinates of the points of the path are transmitted to the server that retrieves the road stretches nearest to the scribble. The stretches are displayed on the map and the user can draw again or delete the undesired ones, always by “rubbing” the mouse over the map. In that intuitive manner the refinement can go on until a suitable set of stretches are selected to describe the bike-tour.

In our CBR-system, a case consists of a set of subjective and objective features. The former are the title of the tour, the approximate duration, the difficulty; the latter include the computed length, the proximity of springs or drinkable water fountain, the altimetric profile, the average slope and so on. Some of the features are symbolic, some are numeric. Among the objective data, the GIS coordinates of the points of the path itself are of central importance.

In order to assess the similarity, the feature-value part of the case presents no problem for a standard Euclidean-Hamming weighted metric is sufficient. The notion of distance on the geographic data of the path must be defined. Different definitions of spatial similarity have been presented in the literature. Usually, the

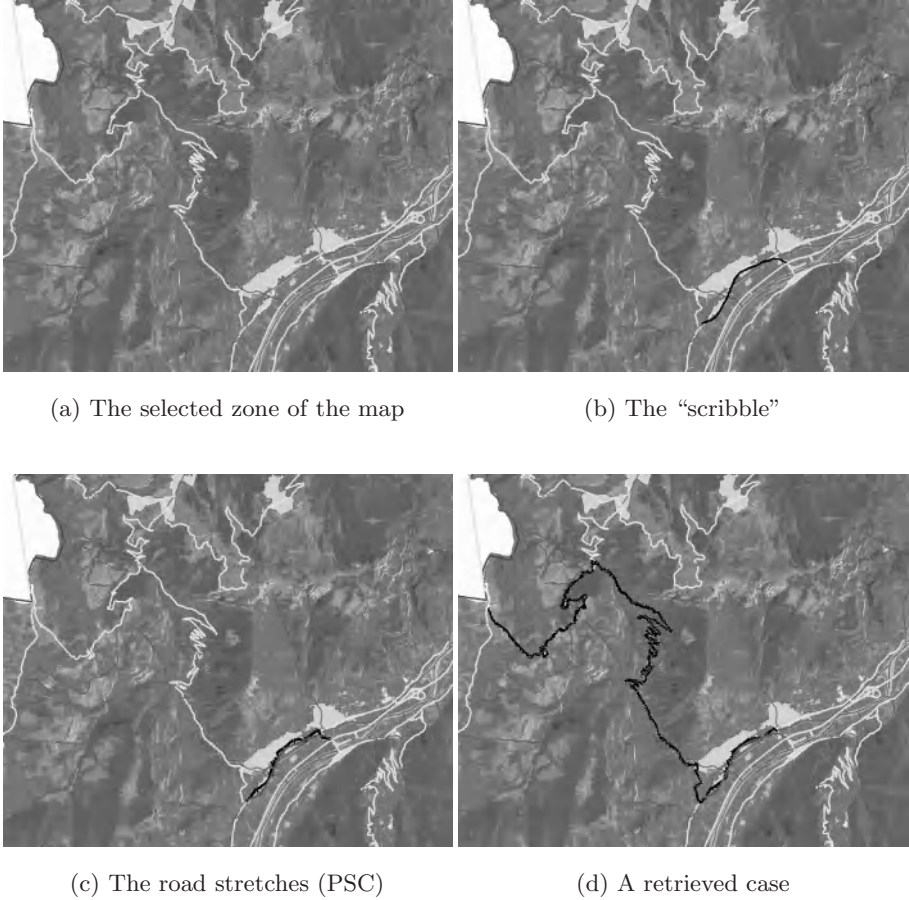


**Fig. 3.** Snapshots of the user interface. The user selects a zone of the map (a), draws a “scribble” using the mouse (b), gets the road stretches near to it (c), and finally a similar tour is retrieved (d)

problem of similarity is addressed at the level of semantics on complex scenes that involve different objects and their spatial relations [13] (see [11] for a review on different approaches). The natural way of defining a distance between compact sets as the Hausdorff metric is revealed to be unsuitable. It requires a quadratic computation on the number of points and even a simple path can consists of hundreds of points. We then adopted a simplified method based on a common way of indexing spatial data. The bounding box of the layer that contains our data is divided in a lattice of  $2^n$  rectangular tiles and a quadrant labeling with a quaternary code is defined in a manner analogous to that explained in [6] (see



Fig. 4). When a sequence of points are stored in the server, the set of tiles over which the path is spanned are recorded in a table along with an identifier for the path itself. The comparison of the spatial data of a tour with another one is then simply done by counting the number of common tiles as a measure of overlap.

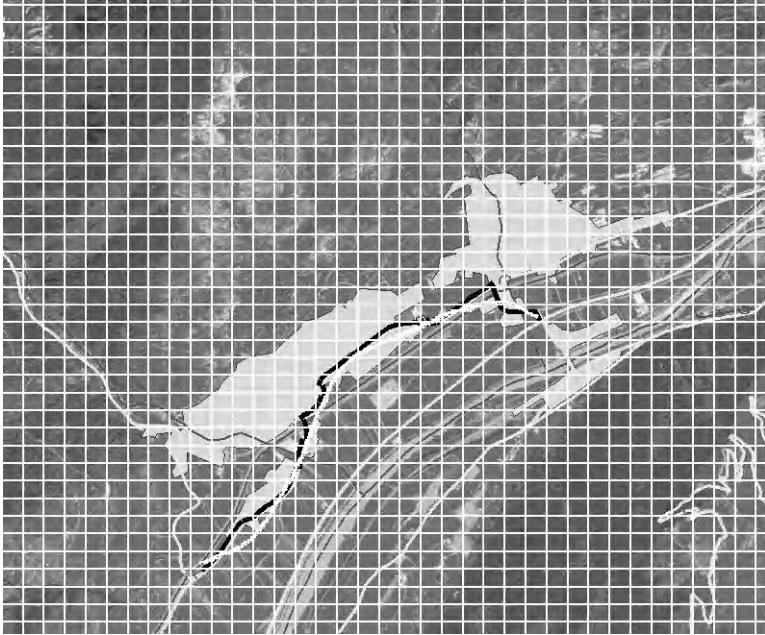


**Fig. 4.** Close-up of the map with the “scribble” (*white*) and the road stretches (*black*) near to it; for sake of clarity the grid used in the matching is superimposed. The number of tiles in common gives the measure of similarity

The system was developed in the framework offered by the Oracle suite of tools. The spatial data, the subjective and user data are stored in an Oracle DataBase (Oracle8 with the Spatial Cartridge extension that supports the quadtree indexing). All the CBR-related *business logic* is hosted on the Oracle Application Server; part of the user-session management is achieved with servlet-like objects written in Java. In this three-tier architecture, the client is a Java-applet that runs in a web-browser: it is a “thin client heavier than it looks”. It manages almost all the *presentation logic*, and it is thin from the deployment point of view. However it takes also care of some of the *business logic*, namely management of user-session data and treatment of tour data. The applet has actually the capability of geo-referencing vectorial spatial data entered by the user or loaded from the server.



A clear visualization is achieved by superimposing these data on a raster that represents the map of the area of interest. The map is created on the fly by a CGI written in VisualBasic based on ESRI's MapObject components.

## 5 Conclusions and Related Work

We have presented COOL–TOUR a CBR system for supporting Tourism Culture, namely building a community of tourists interested in the same area. The system supports the storing of experiences in the form of tours. In particular, the implemented version deals with Mountain Bike Tours, that will be integrated in a tourism site for the promotion of the Provincia Autonoma di Trento. Further work will be required in order to test on the field the application.

The architecture and the functionalities of COOL–TOUR generally addresses the problem of building and maintaining community of tourists via Internet. Case Based Reasoning represents an interesting choice for tourism and, more in general, to all the e-commerce domains where customer-dependent composition of existing products is required.

Given the peculiarity of tourism domain we concentrated on spatial similarity and COOL–TOUR can be seen as an application of CBR on spatial data. CBR were already effectively proposed in conjunction with GIS [10] and spatial similarity has been exploited for analysis of tourist movements [9]. With respect to these systems the innovative contribution of our work is the use of a CBR modality for supporting the culture of the virtual community. No methodical evaluation of the system has been carried on yet. Test on the field will allow to assess the usefulness of this approach.

## Acknowledgements

The authors wish to thank Fausto Giunchiglia, Anna Perini, Paolo Avesani and Stefano Aguzzoli for the useful and stimulating discussions, and the referees for their suggestions.

## References

1. Klaus-Dieter Althoff, Ralph Bergmann, and L. Karl Branting, editors. *Case-Based Reasoning Research and Development*. Springer, Berlin, 1999. 361, 369
2. L. Ardissono and A. Goy. Tailoring the Interaction With Users in Electronic Shops. In Kay [12], pages 35–44. 360
3. D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proceedings of the International Conference on Machine Learning*, pages 46–54, Madison, Wisciautoh, 1998. Morgan Kaufmann Publishers. 360
4. Max J. Egenhofer. Spatial-Query-by-Sketch. In M. Burnett and W. Citrin, editors, *VL'96: IEEE Symposium on Visual Languages*, pages 60–67. Boulder, CO, IEEE Press, 1996. 365

5. B. J. Fogg (editor). Persuasive technologies. *Communications of the ACM*, 42(5):26–58, May 1999. 361
6. Irene Gargantini. An effective way to represent quadrees. *Communications of the ACM*, 25(12):905–910, December 1982. 366
7. C. Ghidini and L. Serafini. Information Integration for Electronic Commerce. In C. Sierra and P. Noriega, editors, *Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET'98)*, volume 1571 of *LNAI*, pages 189–206, Minneapolis, USA, 1998. Springer Verlag. 362
8. Kristian J. Hammond, Robin Burke, and Kathryn Schmitt. A Case-Based Approach to Knowledge Navigation. *Working Papers of the AAAI Workshop on Multi-Media and Artificial Intelligence*, July 1994. Available at <http://www.cs.uchicago.edu/~kris/PDF/nav.pdf>. 358
9. E. C. Higham, A. Holt, and G. W. Kearsley. Tourist flow reasoning: the spatial similarities of tourist movements. In *Proceedings for the Spatial Information Research Centre's 8th Colloquium*, pages 75–83, 1996. 368
10. A. Holt and G. L. Benwell. Applying Case-Based Reasoning Techniques in GIS. *The International Journal of Geographical Information Science*, 13(1):9–25, 1999. 368
11. Alec Holt. The Matching and Ranking of Surface and Deep Similarities of Spatial Data. In *Proceedings of the Spatial Information Research Centre's 10th Colloquium*, pages 133–143, 1998. 366
12. Judy Kay, editor. *User Modeling: Proceedings of the Seventh International Conference, UM99*. Springer, Wien New York, 1999. 368, 369
13. M. Andrea Rodríguez, Max J. Egenhofer, and Robert D. Rugg. Assessing Semantic Similarities among Geospatial Feature Class Definitions. In A. Vckovski, K. Brasel, and H.-J. Schek, editors, *Lectures Notes in Computer Science*, volume 1580, pages 189–202. Springer-Verlag, March 1999. 366
14. Dorian Selz and Petra Schubert. Web Assessment - A Model for the Evaluation and the Assessment of Successful Electronic Commerce Applications. In *Proceedings of the 31st HICSS Conference*, pages 46–48, Hawaii, 1998. 359
15. B. Smyth and P. Cotter. Surfing the Digital Wave: Generating Personalised TV Listings Using Collaborative, Case-Based Recommendation. In Althoff et al. [1], pages 561–571. 358
16. Waszkiewicz, Cunningham, and Byrne. Case-based User Profiling in a Personal Travel Assistant. In Kay [12], pages 323–325. 358
17. Hannes Werthner and Stefan Klein. *Information Technology and Tourism - A Challenging Relationship*. Springer-Verlag, Wien, 1999. 358
18. Wolfgang Wilke, Mario Lenz, and Stefan Wess. Intelligent Sales Support with CBR. In Mario Lenz, Brigitte Bartsch-Spoerl, Hans-Dieter Burkhard, and Stefan Wess, editors, *Case-Based Reasoning Technology - From Foundations to Applications*, *LNAI*, pages 91–113. Springer, Wien New York, 1998. 358, 359
19. Z. Zhang and Q. Yang. Towards lifetime maintenance of case based indexes for continual case based reasoning. In F. Giunchiglia, editor, *Artificial Intelligence: Methodology, Systems and Applications*, pages 489–500. Springer, Berlin, 1998. 358

# A Case-Based Reasoning Approach to Collaborative Filtering

Robin Burke

Department of Information and Computer Science  
University of California, Irvine  
burke@ics.uci.edu

**Abstract.** Collaborative filtering systems make recommendations based on the accumulation of ratings by many users. The process has a case-based reasoning flavor: recommendations are generated by looking at the behavior of other users who are considered similar. However, the features associated with a user are semantically weak compared with those used by CBR systems. This research examines *multi-dimensional* or *semantic* ratings in which a system gets information about the reason behind a preference. Experiments show that metrics in which the semantic meaning of each rating is taken into account have markedly superior performance than simpler techniques.

## 1 Introduction

Collaborative filtering (CF) is a technique for recommending items to a user's attention based on similarities between the past behavior of the user and that of other users. A canonical example is the GroupLens system that recommends news articles based on similarities between users' reading behavior [1]. This technique has been applied to many areas from consumer products to web pages [2,3,4] and has become a standard marketing technique in electronic commerce. CF has some similarities with case-based reasoning, in that specific instances (users) are used to make predictions.

The input to a CF system is a triple consisting of a user, an object that the user has an opinion about, and a rating that captures that opinion:  $\langle u, o, r(u,o) \rangle$ . As ratings for a given user are accumulated, it becomes possible to correlate users with similar ratings and make predictions about unrated items on the basis of historical similarity. In other words, the goal is to find other users  $u_i$  whose ratings correlate well with some user  $u_o$ , and use these users' ratings on some new object  $o'$  to predict  $r(u_o, o')$ .

In case-based reasoning terms, we can consider a collaborative filtering case to consist of the item/rating combinations  $\langle o, r \rangle$  generated by a particular user. A problem situation consists of a case plus an item that the user has not rated. A solution consists of a prediction of the user's likely rating for that item. The research questions for CF primarily center on the problem that in CBR would be called similarity assessment: how to identify users that are similar enough to yield good predictions. Typically, statistical and machine learning approaches have prevailed as opposed to the knowledge-based approaches to similarity assessment found in CBR. Part of the

problem is that the ratings available to CF systems are usually extremely simple: values along a one-dimensional linear scale from like to dislike. With such an impoverished case representation, weak methods are the only ones that can be applied.

Alternatives to collaborative recommendation include content-based and knowledge-based recommendation techniques. Content-based recommendation identifies similar items by comparison of low-level features, such as the words in a news story. Machine learning techniques are employed to derive classifiers to predict which items the user will like. [5,6] Knowledge-based recommendation employs domain knowledge to infer similarities between items, making this class of recommender systems essentially an application of CBR case retrieval.

My research has investigated the creation of knowledge-based recommender systems [7,8] that afford critique-based navigation through product databases. For example, the restaurant guide Entree<sup>1</sup> [9] allows users to navigate through a database by critiquing restaurants as too expensive, or too traditional, among other dimensions. Each critique invokes a new retrieval redirecting the user towards items more likely to meet his or her needs. A critique can be considered a kind of rating, but it is a *semantic* rating. That is, the system knows not only that the user disliked a certain restaurant, but also that the reason was the cost, the cuisine or some other specific factor.

Earlier work [10] noted the potential for integrating collaborative filtering and knowledge-based recommendation and described a framework under which such integration could be accomplished. A “cascading” hybrid recommender system uses its knowledge to generate the best possible set of recommendations, and then uses collaborative filtering to break ties. One of the traditional difficulties with collaborative filtering is the “ramp-up” or “cold-start” problem. When the system has little data about a particular user, correlation with other users does not give good predictions. Similarly, when a new item is added to the database, it will not have been rated by any users and cannot be recommended. A knowledge-based hybrid avoids the ramp-up problem, since the system can make good recommendations without gathering any usage data.

This approach differs from other hybrid recommenders recently proposed. The PTV system [11], which recommends TV programs, presents content-based and collaborative recommendations mixed together in a single personalized schedule. The cascade approach is preferable in our case because of the navigation metaphor. A mixed set of recommendations would not be appropriate because users are browsing in a directed fashion and would be confused by collaborative suggestions, which would generally lack any discernable relation to the user’s most recent critiquing action.

The DailyLearner system [12] uses a hybrid recommender to suggest news stories. It combines content-based and collaborative techniques hierarchically. If one technique cannot make a recommendation for or against a news story with sufficient confidence, the system falls back on the next technique. This approach throws away similarity information returned by one classifier when it moves on to the next, since small differences that do not meet a decision threshold are likely to be noise. By contrast, a knowledge-based hybrid can have more confidence in the judgements of its knowledge-based layers and can build on them in a cascading fashion.

---

<sup>1</sup> <URL: <http://infolab.ils.nwu.edu/entree/>>

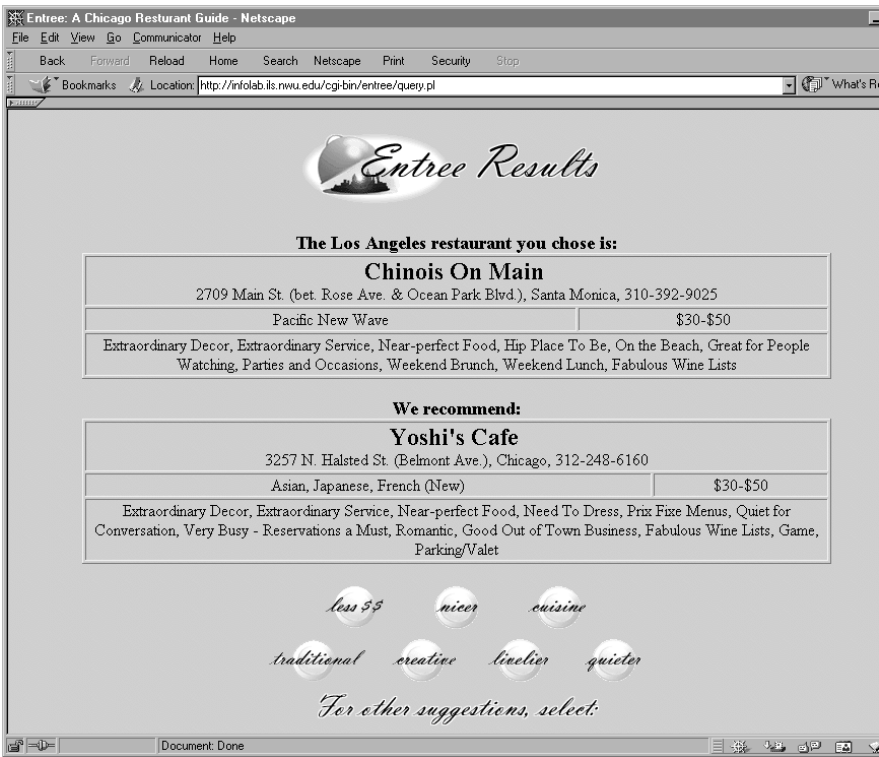


Fig. 1. The Entree recommender system

## 2 Entree

Consider a user who starts browsing using a known restaurant, Wolfgang Puck’s “Chinois on Main” in Los Angeles.<sup>2</sup> As shown in Figure 1, the system finds a similar case, a Chicago restaurant that combines Asian and French influences, “Yoshi’s Cafe,”<sup>3</sup> as well as other restaurants (not shown) that are ranked by their similarity. The user, however, is interested in a cheaper meal and selects the “Less \$\$” button. The result (not shown) is a creative Asian restaurant in a cheaper price bracket: “Lulu’s.” However, the French influence is lost – one consequence of the move to a lower price bracket. The user can continue browsing and critiquing until an acceptable restaurant has been located.

<sup>2</sup> The user may also make a database query based on desired restaurant features.

<sup>3</sup> Note that the connection between “Pacific New Wave” cuisine and its Asian and French culinary components is part of the system’s knowledge base of cuisines.

Entree has been in continuous operation as a public web utility since July 1996. The experiments described below use logs through June 1999. The system does not retain any user data – it does not use cookies or other mechanisms to identify a user returning to the site. For this reason, these experiments may show something of a lower bound on the efficacy of CF: a site would normally be expected to gather user ratings over multiple sessions of interaction and tie them directly to a user identifier.<sup>4</sup> The log data was partitioned into sessions, identified by IP address and terminating after 10 minutes of inactivity. There are approximately 50,000 sessions in the 3 years of usage data.<sup>5</sup>

Each session consists of a list of user actions and displayed restaurants. As shown in the screen shot, a number of restaurants are retrieved as likely candidates, but only one is highlighted. There are eight possible actions a user can take: “Less \$\$,” “Nicer,” “More Creative,” “More Traditional,” “Quieter,” “Livelier,” “Change Cuisine,” and “Browse” (the choice to move to a different restaurant in the return list.) In the case of the user choosing an alternative cuisine, the log does not record the user’s choice. A user can begin the session with a known restaurant as a starting point or with a query that describes the type of restaurant sought. (These queries were also not logged.) So, for each session, we can associate one of 10 ratings: Entry point, Exit point, and the eight actions.

The sessions range in length from one to 20 interactions, but typically contain less than ten actions. Frequently, the same restaurant is rated more than once. For example, a user might see a recommendation, browse to the other restaurants in the list, return to the original suggestion, and then perform a tweaking action. We discard all but the most recent rating for each restaurant.

The research reported here attempts to estimate how much improvement might be expected from using the cascading hybrid with Entree, and determine what algorithm would produce the best performance for the collaborative component. The central issues are the generation of ratings and the computation of intra-user similarity. A standard collaborative filtering approach would be to project the multi-dimensional ratings onto a single dimension, a simple binary like / dislike scale. For example, if user A sees “Yoshi’s Cafe” and clicks “Less \$\$,” this can be recorded as a negative rating. If the user stops browsing upon encountering this restaurant, the user is assumed to have found what he or she seeks, and a positive rating is assigned.<sup>6</sup> Ratings produced in this way can be plugged directly into standard CF algorithms.

Another approach is to look only for users with exactly the same ratings: treating ratings on different scales as incommensurable. We would only match user A against others who also thought that “Yoshi’s Cafe” was too expensive. We can formulate the problem as an information retrieval task where each term consists of a restaurant / rating combination. For example, we can use a cosine measure of similarity, treating each user as a multi-dimensional vector. The drawback of this “sparse” metric is that

<sup>4</sup> On the other hand, ratings gathered over a longer period of time would reflect a diversity of search goals, a diversity presumably not present in a single search session.

<sup>5</sup> The Entree data set is available at the UCI KDD Archive <URL: <http://kdd.ics.uci.edu/> >.

<sup>6</sup> It is possible that the user has given up in frustration, in which case a positive rating would be inappropriate. This kind of ambiguity is common in other CF domains, such as web log analysis. A system that was tied to e-commerce transactions or in the case of restaurants, on-line reservation placement, would have more reliable positive ratings.

Table 1. Adjacency matrix for Entree ratings

| Br | Ch. | Ni. | Tr   | Cr.  | Li.  | Qu.  | Cu. | En. | Ex |         |
|----|-----|-----|------|------|------|------|-----|-----|----|---------|
| 1  | 0   | 0   | 0    | 0    | 0    | 0    | 0   | 0   | 0  | Browse  |
|    | 1   | -1  | -0.5 | -0.5 | -0.5 | -0.5 | 0   | 0   | 0  | Cheaper |
|    |     | 1   | 0.5  | 0.5  | -0.5 | 0.5  | 0   | 0   | 0  | Nicer   |
|    |     |     | 1    | -1   | -0.5 | 0.5  | 0   | 0   | 0  | Trad.   |
|    |     |     |      | 1    | 0.5  | -0.5 | 0   | 0   | 0  | Creat.  |
|    |     |     |      |      | 1    | -1   | 0   | 0   | 0  | Lively  |
|    |     |     |      |      |      | 1    | 0   | 0   | 0  | Quiet   |
|    |     |     |      |      |      |      | 1   | 0   | 0  | Cuisine |
|    |     |     |      |      |      |      |     | 1   | 1  | Entry   |
|    |     |     |      |      |      |      |     |     | 1  | Exit    |

only a small number of users would have had exactly the same reactions to a given set of restaurants, meaning that predictions will be based on a smaller number of users than in a collaborative approach.

A third technique takes into account the semantics of the ratings themselves: we establish a similarity metric between ratings based on their characteristics. For example, if user B looks at “Yoshi’s Café” and clicks “Nicer,” we should probably rate users A and B as dissimilar even though they both disliked the same restaurant – they did so for essentially opposite reasons. This is a “heuristic similarity” approach, a familiar idea in a CBR context. It does not establish a single scale onto which all ratings are projected, but rather looks at the similarity of users on a rating by rating basis. This metric takes the qualitative differences between ratings into account, but it allows more kinds of intra-user comparison than the sparse metric. A similarity value is assigned to each possible pair of ratings, using an adjacency table generated by considering the semantics of each response type, and a few common-sense considerations:

- A rating is maximally similar to itself.
- “Browse” is not similar to any other rating.
- Some ratings have natural opposites: “Livelier” / “Quieter”, “Traditional” / “Creative.”

The full comparison table is shown in Table 1. Similarity between users is determined by averaging their scores for all restaurants rated in common.

### 3 Evaluation

To evaluate the different CF approaches, the data was partitioned into equal-sized training and test sets of approximately 25,000 sessions. From the test set, only highly active users were extracted, those with at least 15 restaurants rated, 199 users in total. These were the only users who provided enough data to perform the evaluation, but they are a small minority of the test data. A commercial system that used cookies to identify returning users would accumulate 15 ratings for a user very quickly.

The goal of CF for Entree is not strictly to predict ratings, but rather to improve the quality of the recommendations made by the knowledge-based component. The evaluation technique reflects this application. See the outline of the algorithm in Figure 2. For each session  $S$ , the system first isolates a positively rated restaurant  $r$  – an item the user found satisfactory. The goal is to bring this restaurant to the user’s attention as soon as possible, so the evaluation task for the CF system is to pick this positively rated item from a group of negatively rated ones. To simulate this situation, the system randomly selects 6 items from  $S$  with a negative rating, and groups them with  $r$  as the test partition  $T$ . Eight items then become training data for a particular user, excess ratings being discarded at random. (Five such training/test splits are generated for each session for cross-validation.) Using the training partition, a prediction is made for the rating of each test item  $t \in T$ , and the one with the highest predicted score is selected as the recommendation. Additional training data is added in steps, and predictive performance is reported when four, six or eight of the ratings are known to the system.

For the correlation filter, predictions of the rating of a test item  $t$  are made by selecting all users who have rated  $t$ , selecting those who meet a minimum threshold of correlation with the test user based on the training data, and averaging their ratings of  $t$ .<sup>7</sup> The same experimental scheme was applied also to the sparse metric, using cosine similarity, and with the heuristic metric, using the average of the adjacency values for all items rated in common. As a baseline, we also used a predictor based on the average rating of a restaurant across all users.

The experiment also evaluated two different ways of projecting ratings on a scale: a binary method and a scalar method. Under the binary method, most ratings have a score  $-1$  (since they cause the user to move away from the restaurant in question). The only exceptions are “Entry point” and “Exit point,” which are both treated as positive ratings ( $+1$ ). The scalar method adds two additional distinctions: “Exit point” gets a

```

Let S = A session: {  $s_0, \dots, s_n$  },  $n \geq 15$  where each  $s_i$ 
    consists of a pair <restaurant, rating>
r = a positive rating from S
T = test data for the session, initially { }
P(S, t), a prediction function that predicts the
    score of test item t, given ratings from
    training data in S.

move r from S to T
move 6 random s from S to T
for i <- 4, 6, 8
    p <- t such that P({ $s_0, \dots, s_i$ }, t) is maximized
    if p equals r
        correct prediction (i)

```

**Fig. 2.** Evaluation algorithm

<sup>7</sup> Since the rating data is relatively sparse, a default score of 0 (no preference) is assigned when an item is rated in one session but not in the other [13].



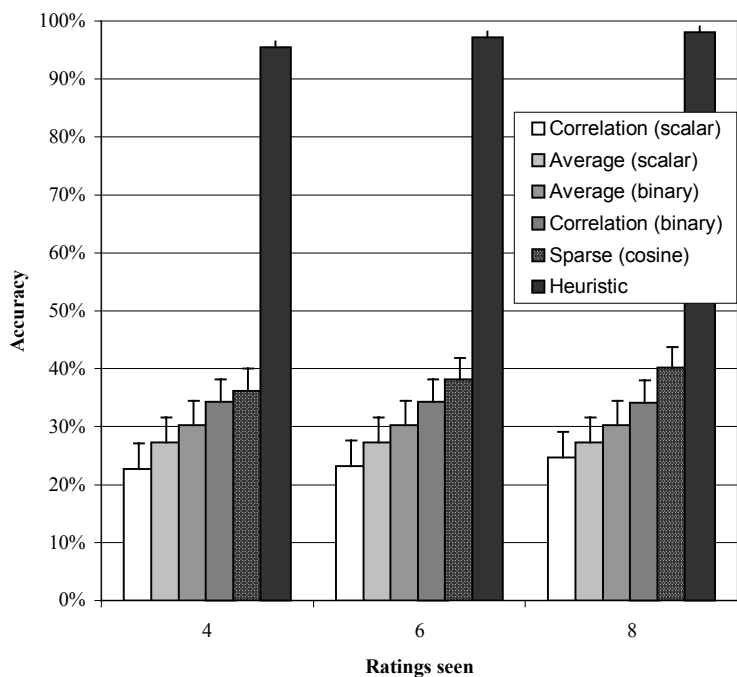


**Fig. 3.** Experimental results

slightly lower rating (0.8) since we cannot be fully confident that it signifies success as opposed to frustration, and “Browse” gets less of a negative rating (-0.5) to reflect the fact that users are not directly critiquing a restaurant, but instead poking around among other returned items. The hypothesis was that the scalar technique, by incorporating some semantic information about the meaning of ratings, would have performance in between the correlation and heuristic techniques.

The experiment therefore had six conditions: Correlation computed with both binary and scalar ratings, the Sparse metric using cosine similarity, the Heuristic metric, together with a pair of predictors using the Average rating computed with the two rating techniques. Figure 3 shows the results of the experiment. (Error bars represent the 95% confidence interval.) Surprisingly, the scalar version of the ratings was not better than the simpler binary projection. Both the Average and Correlation metrics using scalar ratings were slightly worse than the Average using binary ratings. The Sparse metric slightly outperformed Correlation using binary ratings, but this difference is not statistically significant. However, the Heuristic technique was shown to be the clear winner. It approached 100% accuracy at the task of predicting what restaurant the user will like, and even with a small amount of data (4 ratings), better than 95% accuracy was achieved.

The performance of the Heuristic technique was further evaluated given a smaller amount of training data. A sample of the data (6600 sessions) was extracted and performance evaluated on shorter sessions (10 ratings or more, total of 264 users), using five ratings for testing and a maximum of five ratings for training. Figure 4

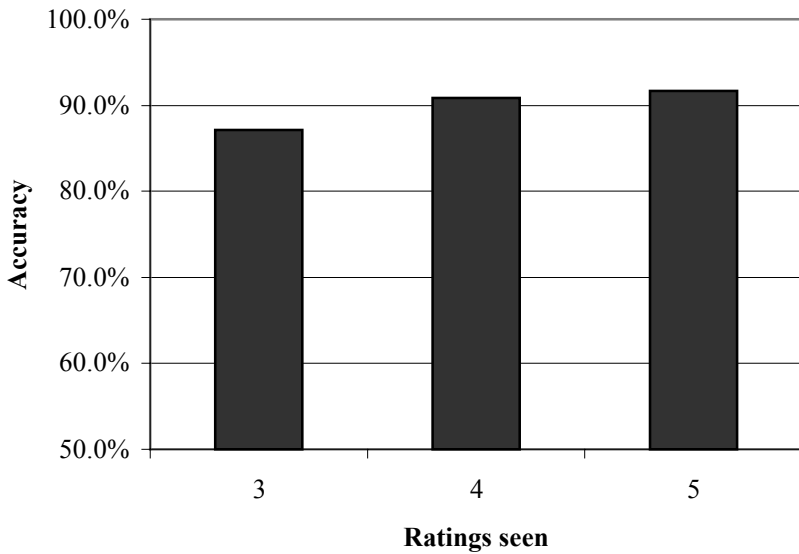


**Fig. 4.** Heuristic prediction for users with 10 ratings or more

shows these results, which are consistent with the performance for more active users. Having seen three ratings, the Heuristic metric can predict the preferred item from the five item test set with 87% accuracy, rising to 92% after all five ratings are seen.

These experiments show that the heuristic metric is an extremely good predictor of user tastes in this data. None of the other techniques evaluated is even half as accurate. More sophisticated statistical and machine learning techniques have been applied that improve on the correlation approach used here. However, the differences in performance are much smaller than the gains shown here by the heuristic metric. This can be attributed to a “rubber ceiling” effect – with limited data, ever more powerful techniques must be applied to squeeze out small performance gains. Using just a small amount of semantic information, identifying the reason behind a user’s preference, a case-based approach can break through this ceiling to a new level of accuracy.

## 4 Future Work

The main drawback to the heuristic approach is that it involves additional knowledge engineering: for  $n$  possible critiques, an  $O(n^2)$  adjacency matrix must be generated. This is particularly a problem in domains that have a large number of possible critiques, such as the domain of movies. A user might, for example, ask for a movie that is similar to one being displayed but without a particular actor: “Die Hard” without Bruce Willis. It would be impractical to manually construct an adjacency matrix rating the similarity of all such choices. Further research will investigate whether the similarity between two tweaks can be determined by examining their behavior, looking at the differences between the starting point and the items retrieved

by the tweak. For example, a tweak that removed a particular feature should be considered different from one that added the same feature. Another possibility is to use unsupervised learning to determine redundancies in the whole body of rating data and establish inter-rating similarities on that basis. Latent semantic indexing is one possible technique that could be employed for this purpose. Another area of active investigation is the clustering and categorization of users based on qualitative ratings.

The heuristic approach to collaborative filtering described above is being applied at the commercial successor to Entree: Recommender, Inc., a provider of recommendation generation services to e-commerce sites.

## 5 Conclusion

The experiments described here show that semantic ratings can be very powerful contributors to collaborative filtering. It is hardly surprising that the meaning of actions should be an important consideration in determining similarity between user profiles: why a user likes or dislikes something must surely be important. Nor is it surprising that heuristic similarity has received little attention in collaborative filtering research, given that not many applications have access to semantic critiques such as those used in Entree. However, our results demonstrate that with semantic ratings, a case-based approach to collaborative filtering is very successful, far superior to knowledge-weak approaches. Perhaps the most effective research direction for recommender systems is therefore not the pursuit of better analysis of simple rating data, but the challenge of designing user interfaces that capture the semantics as well as the statistics of user preference.

## Acknowledgements

These experiments were designed with the assistance of Daniel Billsus at UCI. Entree was developed at the University of Chicago in collaboration with Kristian Hammond, with the support of the Office of Naval Research under grant F49620-88-D-0058. The interface to the Entree system was designed and created by Robin Hunicke. Many others contributed to the FindMe effort at the University of Chicago, including Terrence Asselin, Kai Martin, Kass Schmitt, and Robb Thomas.

## References

- [1] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the conference on Computer supported cooperative work*, pp. 175-186. New York: ACM Press.
- [2] Shardanand, U. and Maes, P. 1995. Social information filtering algorithms for automating "word of mouth" In *CHI-95: Conference proceedings on Human factors in computing systems*, pp. 210-217. New York: ACM Press.

- [3] Resnick, P. and Varian, H. R. 1997. Recommender systems. *Communications of the ACM*, 40(3) 56-58.
- [4] Kautz, H. (ed.) 1998 *Recommender Systems: Papers from the AAAI Workshop*. AAAI Technical Report WS-98-08. AAAI.
- [5] Balabanovic, M. and Shoham, Y. 1997. FAB: Content-Based Collaborative Recommender. . *Communications of the ACM*, 40(3) 66-72.
- [6] Pazzani, M. and Billsus, D. 1997. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning* 27(3): 313-331.
- [7] Burke, R. 1999 The Wasabi Personal Shopper: A Case-Based Recommender System. In *Proceedings of the 11th National Conference on Innovative Applications of Artificial Intelligence*, pp. 844-849, AAAI.
- [8] Burke, R. In press. Knowledge-based Recommender Systems. In A. Kent (ed.), *Encyclopedia of Library and Information Systems*.
- [9] Burke, R., Hammond, K., and Young, B. 1997 The FindMe Approach to Assisted Browsing. *IEEE Expert*, 12(4), 32-40.
- [10] Burke, R. 1999 Integrating Knowledge-Based and Collaborative-Filtering Recommender Systems. In *AAAI Workshop on AI in Electronic Commerce*. pp. 69-72. AAAI.
- [11] Cotter, P. & Smyth, B. 2000. PTV: Intelligent Personalised TV Guides. In *Proceedings of the 12th Innovative Applications of Artificial Intelligence Conference*. AAAI Press. To appear.
- [12] Billsus, D. and Pazzani, M. 1999. A Hybrid User Model for News Story Classification. In *Proceedings of the Seventh International Conference on User Modeling (UM '99)*, Banff, Canada, June 20-24, 1999.
- [13] Breese, J. S.; Heckerman, D. and Kadie, C. 1998. Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43-52. San Francisco, CA: Morgan Kaufmann.

# Similarity Measures for Structured Representations: A Definitional Approach

Göran Falkman<sup>1,2</sup>

<sup>1</sup> Department of Computing Science,  
Chalmers University of Technology and Göteborg University  
SE-412 96 Göteborg, Sweden

<sup>2</sup> Department of Computer Science, University of Skövde  
PO Box 408, SE-541 28 Skövde, Sweden  
`goran.falkman@ida.his.se`

**Abstract.** A similarity framework for definitional representations is presented. Similarity assessment is based on the computation and estimation of structural relationships (connections) among definitions. The framework is general enough to capture many different types of similarity measures: ordinal and cardinal measures, asymmetric measures, and measures between any number of objects. By definition, the similarity measures retrieve the cases that need the least adaptation.

## 1 Introduction

Similarity assessments play an important role in most cognitive activities. Clinical medicine is not an exception to this. On the contrary, a clinician examining a patient wants to know if there are previous examination records that are similar to the current one, hoping that these might be helpful in the diagnosis of the new patient.

Case-based reasoning (CBR) [5], is a problem solving method that is appropriate for this kind of problem. The main idea in CBR is to use the solution of a problem that has been solved earlier to solve a new problem. To do this, the similarity between the new problem and each of the previous problems must be computed. The solution of the case that is most similar to the new problem is then adapted in order to construct a solution to the new problem.

In most CBR applications, past experiences have no inherent structure and are represented as tuples of attribute-value pairs. Traditionally, similarity is assessed by a numeric computation, as a weighted sum of the similarity between individual attribute values, and results in a single numeric value.

In contrast to this, there are more structured approaches to CBR where cases are not given by collections of attribute-value pairs but by structured representations. Examples of structures that have been used are graphs [2], trees [10], and nested terms [8,9]. In this approach, similarity assessment is based on structural similarity, where the similarity between two structured cases is captured by

structural relations that are common to both cases. Usually, structural similarity provides not only a numeric value but also the most specific structure two cases have in common.

In 1995, the MedView project, based on a co-operation between computing science and clinical medicine, was initiated. The overall goal of the project was to develop models, methods, and tools to support clinicians in their daily diagnostic work. In MedView, examination records are given as definitional structures [3]. Thus, to be able to use previous clinical knowledge in solving new problems, we must first study how to measure the similarity between definitions.

This paper adds to the ongoing research within the MedView project by presenting a framework for similarity assessment for definitional representations. This work also contributes to research on structured representations and research on structural similarity in CBR.

The framework is general enough to capture many different types of similarity measures: ordinal and cardinal measures, asymmetric measures, and measures between any number of objects. By definition, the similarity measures retrieve the cases that need the least adaptation.

The paper is organised as follows: The next section gives a short introduction to definitions as data structures. In the following section we present our similarity framework for definitional representations. The paper is concluded with a general discussion.

## 2 Definitions as Data Structures

Definitional structures [3] have been used as data structures, as a basis for declarative programming, and as a knowledge representation model in both engineering, construction planning, clinical odontology, and bioinformatics.

In this section we present the basic notions: definitions, methods, and computations. We conclude the section with a small example.

### 2.1 Definitions

Intuitively, a *definition* defines possible connections between objects in a universe of objects. Formally, a definition  $\mathcal{D}$  is given by

1. two sets: the *domain* and the *co-domain*, where  $\text{Dom}(\mathcal{D}) \subseteq \text{Com}(\mathcal{D})$ ,
2. and a *definiens* operation:  $\mathcal{D} : \text{Com}(\mathcal{D}) \rightarrow \mathcal{P}(\text{Com}(\mathcal{D}))$ .

The conceptual view of a definition is that of a system of equations, where the left-hand sides (*atoms*) are defined in terms of the right-hand sides (*conditions*):

$$\mathcal{D} \left\{ \begin{array}{l} a_0 = A_0 \\ a_1 = A_1 \\ \vdots \\ a_n = A_n \end{array} \right. \quad n \geq 0,$$

where  $a_i \in \text{Dom}(\mathcal{D})$  and  $A_i \in \text{Com}(\mathcal{D})$ ,  $0 \leq i \leq n$ . Each  $A_i$  may in turn be a definition. Thus, definitions can be represented by nested systems of equations.

The intuitive interpretation of the definiens operation is that it gives further information about its argument. The definiens of an atom  $a$  in a definition  $\mathcal{D}$  gives the conditions defining  $a$  in  $\mathcal{D}$ , i.e.,  $\mathcal{D}(a) = \{A \mid (a = A) \in \mathcal{D}\}$ . The definiens of a condition  $A$  is the set of the constituents of  $A$ .

We let  $(A/a)\mathcal{D}$  be the result of substituting all left-hand sides  $a$  in  $\mathcal{D}$  with  $A$ . Similarly,  $\mathcal{D}(B/b)$  is the result of substituting all right-hand sides  $b$  in  $\mathcal{D}$  with  $B$ .

We let  ${}_e\mathcal{D}$  denote the left-hand side of some equation  $e \in \mathcal{D}$ . Similarly,  $\mathcal{D}_e$  denotes the right-hand side of some equation  $e \in \mathcal{D}$ .

## 2.2 Methods

Let  $\mathcal{V}$  be a set of computation variables. Let  $\mathcal{O}$  be a set of formal notations  $\overline{\mathcal{D}}$  and  $\underline{\mathcal{D}}$  for all definitions  $\mathcal{D}$  in some set of definitions. Let  $\mathcal{W} = (\mathcal{V} \cup \mathcal{O})^*$  be the set of words over  $\mathcal{V}$  and  $\mathcal{O}$  (the set of sequences constructed from elements of  $\mathcal{V}$  and  $\mathcal{O}$ ). The empty word is denoted by  $\epsilon$ .

A *computation method* is a definition with  $\mathcal{V}$  as its domain and  $\mathcal{W}$  as its co-domain.

With each method  $\alpha$  we associate a boolean function  $g : \mathcal{O} \times \mathcal{P}(\mathcal{O}) \times \mathcal{E} \rightarrow \text{Bool}$ , for some definition  $\mathcal{E}$ . The function  $g$  acts like a *guard*, by imposing additional constraints on the definiens operation in  $\alpha$ :

$$\alpha_{\mathcal{E}}(v) = \{W \in \alpha(v) \mid g(W, \alpha(v), \mathcal{E}) = \top\}.$$

Thus, a method can be viewed as a system of guarded equations.

The following is an example of a typical method:

$$\alpha_{\mathcal{L}, \mathcal{R}} \begin{cases} \alpha = \alpha \overline{\mathcal{L}} & \text{if } {}_e\mathcal{E} \in \text{Dom}(\mathcal{L}) \\ \alpha = \alpha \underline{\mathcal{R}} & \text{if } \mathcal{E}_e \in \text{Dom}(\mathcal{R}) \wedge {}_e\mathcal{E} \not\in \text{Dom}(\mathcal{L}) \\ \alpha = \epsilon & \text{otherwise} \end{cases} \quad (1)$$

If  $\mathcal{E}$  is the definition to which  $\alpha_{\mathcal{L}, \mathcal{R}}$  is applied, we may interpret (1) as follows: If the left-hand side of some equation in  $\mathcal{E}$  is defined in  $\mathcal{L}$  then replace the left-hand side with its definition in  $\mathcal{L}$ . Otherwise, if the right-hand side of some equation in  $\mathcal{E}$  is defined in  $\mathcal{R}$  then replace the right-hand side with one of its definitions in  $\mathcal{R}$ . Otherwise, end the computation. Thus, we repeatedly replace the left- or right-hand side of some equation in  $\mathcal{E}$  with its definition in  $\mathcal{L}$  or  $\mathcal{R}$  respectively, until nothing more in  $\mathcal{E}$  is defined in  $\mathcal{L}$  or  $\mathcal{R}$ .

For the sake of simplicity we assume that each method is defined in a definition with the same name as the method. That is, the only atom defined in a method named  $\alpha$  is  $\alpha$ .

## 2.3 Computations

We compute a method by applying it to an initial definition. The result of a computation is a new definition. The result definition contains connections between atoms as well as information about how the connections were computed.

$$\mathcal{M}_1 \left\{ \begin{array}{l} \text{disease} = \text{Herpes labialis} \\ \text{mucos-site} = \text{l735} \\ \text{mucos-col} = \text{white} \\ \text{mucos-col} = \text{brown} \\ \text{mucos-txt} = \text{plaque} \\ \text{palp-site} = \text{l735} \end{array} \right. \quad \mathcal{M}_2 \left\{ \begin{array}{l} \text{disease} = \text{Shingles} \\ \text{mucos-site} = \text{l122} \\ \text{mucos-col} = \text{white} \\ \text{mucos-txt} = \text{ulcus} \\ \text{palp-site} = \text{l122} \end{array} \right. \quad \mathcal{M}_3 \left\{ \begin{array}{l} \text{disease} = \text{Diabetes} \\ \text{mucos-site} = \text{l735} \\ \text{mucos-col} = \text{red} \\ \text{mucos-txt} = \text{striae} \\ \text{palp-site} = \text{l737} \end{array} \right.$$

**Fig. 1.** Three partial cases in MedView

The operational semantics is given by the following calculus. A goal in the calculus is a triple,  $\langle W, \mathcal{E}, \mathcal{E}' \rangle$ , where  $W$  is a word and  $\mathcal{E}$  and  $\mathcal{E}'$  are definitions.

1. *Termination*

$$\overline{\langle \epsilon, \mathcal{E}, \mathcal{E}' \rangle}.$$

2. *Method*

$$\frac{\langle WW_1, \mathcal{E}, \mathcal{E}_1 \rangle \cdots \langle WW_n, \mathcal{E}, \mathcal{E}_n \rangle}{\langle W\alpha, \mathcal{E}, \mathcal{E}' \rangle},$$

where  $\alpha_{\mathcal{E}}(\alpha) = \{W_1, \dots, W_n\}$  and  $\mathcal{E}' = \{W_1 = \mathcal{E}_1, \dots, W_n = \mathcal{E}_n\}$ .

3. *Domain Left*

$$\frac{\langle W, (A_1/a)\mathcal{E}, \mathcal{E}_1 \rangle \cdots \langle W, (A_n/a)\mathcal{E}, \mathcal{E}_n \rangle}{\langle W\overline{\mathcal{D}}, \mathcal{E}, \mathcal{E}' \rangle},$$

where  $\mathcal{D}(a) = \{A_1, \dots, A_n\}$  and  $\mathcal{E}' = \{(a = A_1) = \mathcal{E}_1, \dots, (a = A_n) = \mathcal{E}_n\}$ .

4. *Domain Right*

$$\frac{\langle W, \mathcal{E}^{(B/b)}, \mathcal{E}' \rangle}{\langle W\underline{\mathcal{D}}, \mathcal{E}, \mathcal{E}' \rangle},$$

where  $B \in \mathcal{D}(b)$ . Note that this rule introduces a non-deterministic choice between possible results.

## 2.4 An Example

Suppose we want to compare the cases  $\mathcal{M}_1$  and  $\mathcal{M}_2$  in Fig. 1 with respect to the attribute *mucos-col*. This is captured by defining an initial definition  $\mathcal{E} = \{\text{mucos-col} = \text{mucos-col}\}$ . We then apply (1) to  $\mathcal{M}_1$ ,  $\mathcal{M}_2$ , and  $\mathcal{E}$ . In the first step of the computation,  $\mathcal{E}$  is transformed into  $\{\text{white} = \text{mucos-col}, \text{brown} = \text{mucos-col}\}$  using the first equation of (1). After two applications of the second



equation of (1)  $\mathcal{E}$  becomes  $\{white = white, brown = white\}$ . Since nothing more in  $\mathcal{E}$  is defined in  $\mathcal{M}_1$  or  $\mathcal{M}_2$ , the computation ends with the result definition:

$$\alpha_{\mathcal{M}_1, \mathcal{M}_2} \mathcal{E} = \left\{ \overline{\alpha \mathcal{M}_1} = \begin{cases} (mucos-col = white) = \{ \alpha \mathcal{M}_2 = \{ white = white \\ (mucos-col = brown) = \{ \alpha \mathcal{M}_2 = \{ brown = white \end{cases} \right.$$

The result tells us that only one value of *mucos-col* in  $\mathcal{M}_1$  is also a value of *mucos-col* in  $\mathcal{M}_2$ . In other words,  $\mathcal{M}_1$  has only one of its colors in common with  $\mathcal{M}_2$ . We may interpret this as  $\mathcal{M}_1$  not being similar to  $\mathcal{M}_2$ .

In most cases, we are only interested in the final equations, and not in the complete structure of the result. Thus, the result above can be simplified to  $\{white = white, brown = white\}$ . In the following we will assume that all result definitions have been *flattened* in this manner.

### 3 Similarity Measures for Definitional Representations

In this section we present our similarity framework for definitional representations. Both basic and more complex measures are presented.

#### 3.1 Similarity Measures

Computation results describe both structural and computational similarities and differences among definitions. Our approach for measuring the similarity between definitions is based on the idea of using computation results as the basis for similarity assessment.

**Definition 1 (Similarity Measure).** *Let  $\mathcal{D}_1, \dots, \mathcal{D}_n$  be a number of definitions and let  $\alpha$  be a method. Further, let  $\mathcal{E}$  be a system of equations constructed from elements in  $\bigcup_{i=1}^n \text{Com}(\mathcal{D}_i)$ . Then*

$$\alpha_{\mathcal{D}_1, \dots, \mathcal{D}_n} \mathcal{E} \subseteq \{\mathcal{X}_e\}_{e \in \mathcal{E}}$$

*gives the connections between  $\mathcal{D}_1, \dots, \mathcal{D}_n$ .*

*The result of computing  $\alpha_{\mathcal{D}_1, \dots, \mathcal{D}_n} \mathcal{E}$ , which we call a connecting definition, depends on  $\alpha$  and  $\mathcal{E}$  (and also on  $\mathcal{D}_1, \dots, \mathcal{D}_n$ ) and we get different connecting definitions for different types of methods and initial definitions. Therefore, we say that  $\alpha$  and  $\mathcal{E}$  together define a similarity measure.*

The connecting definitions can not only be used as a basis for similarity assessment, but also for measuring dissimilarity or other relations between definitions, e.g., distance.<sup>1</sup>

One may think of the initial definition as a set of test points, a number of properties of the definitions on which the similarity measure is based. The initial

---

<sup>1</sup> We use the term “similarity measure” since the intended use of the connecting definitions in this paper is for similarity assessment.

definition defines equations that we want to solve, connections between atoms in the domain that we want to establish. The computation method is simply the algorithm we use to compare the definitions. The connecting definition defines the equations that must already be solved in order to solve the equations defined in the initial definition, connections between atoms in the domain that must already exist in order to establish the connections in the initial definition.

This idea is similar to the unification of first order terms. The result of a unification, the computed unifier, consists of substitutions that make the terms equal. Let  $X, Y, Z, \dots$  denote variables. Then  $\mathcal{T}_1$  and  $\mathcal{T}_2$  define the two terms  $h(1, X)$  and  $h(1, 2)$  with functor  $h$  and two arguments:

$$\mathcal{T}_1 \left\{ \begin{array}{l} f = h \\ a_1 = 1 \\ a_2 = X \end{array} \right. \quad \mathcal{T}_2 \left\{ \begin{array}{l} f = h \\ a_1 = 1 \\ a_2 = 2 \end{array} \right. .$$

If we apply (1) to  $\mathcal{T}_1$ ,  $\mathcal{T}_2$ , and  $\mathcal{E} = \{f = f, a_1 = a_1, a_2 = a_2\}$  we get

$$\mathcal{U} \left\{ \begin{array}{l} h = h \\ 1 = 1 \\ X = 2 \end{array} \right. ,$$

which we interpret as  $\mathcal{T}_1$  and  $\mathcal{T}_2$  being unifiable. Disregarding the identities,  $\mathcal{U}$  is the unifier of  $\mathcal{T}_1$  and  $\mathcal{T}_2$ .

### 3.2 Basic Measures

Without domain knowledge, i.e., knowledge about the domains and the structure of the definitions, how can we measure the similarity between definitions?

Even if we do not know anything about the atoms in the domain, we could at least ask if there is a connection between an atom in one definition and the same atom in another definition. If we do not know anything about the structure of the definitions, all we can do is to explore the whole structure. Even without any domain knowledge, it is reasonable to assume that there is a connection between every atom and itself. This natural base case is captured by the following definition.

**Definition 2 (Connects).** Let  $\mathcal{A}$  and  $\mathcal{B}$  be two definitions. Further, let  $\mathcal{I} = \{a = a \mid a \in \text{Com}(\mathcal{A}) \cup \text{Com}(\mathcal{B})\}$ , let  $\mathcal{E} \subseteq \{C = C \mid C \in \text{Com}(\mathcal{A}) \cup \text{Com}(\mathcal{B})\}$ , and let  $\alpha$  be the method in (1). If

$$\alpha_{\mathcal{A}, \mathcal{B}} \mathcal{E} = \mathcal{X} \subseteq \mathcal{I}$$

we say that  $\mathcal{A}$  connects to  $\mathcal{B}$  (through  $\mathcal{X}$ ).

*Example 1 (Pattern Matching).* Let  $\mathcal{M}_1$  and  $\mathcal{M}_2$  be the first two examination records in Fig. 1. Suppose we want to find all records with *mucos-col* = *white* and *mucos-txt* = *plaque*. We can then define the following query:

$$\mathcal{Q} \left\{ \begin{array}{l} \text{mucos-col} = \text{white} \\ \text{mucos-txt} = \text{plaque} \end{array} \right. .$$

When comparing our query with a record, we want to compare all attributes in the query with the corresponding attributes in the record:

$$\mathcal{E} \begin{cases} mucos-col = mucos-col \\ mucos-txt = mucos-txt \end{cases}.$$

We may interpret  $\mathcal{E}$  as if we define the similarity between the query and a record in terms of the similarities between the attributes *mucos-col* and *mucos-txt*.

If we compute  $\alpha_{\mathcal{Q}, \mathcal{M}_1} \mathcal{E}$  we get two possible answers:

$$\begin{cases} white = white \\ plaque = plaque \end{cases} \quad \begin{cases} white = brown \\ plaque = plaque \end{cases}.$$

The result of computing  $\alpha_{\mathcal{Q}, \mathcal{M}_2} \mathcal{E}$  is  $\{white = white, plaque = ulcus\}$ . If we require the values of all attributes in the query to be identical to the values of the corresponding attributes in the record, we see that  $\mathcal{M}_1$  is a valid answer to our query, but  $\mathcal{M}_2$  is not.

In this example,  $\mathcal{Q}$  is used as a *pattern*, which is *matched* against the records in the case base.  $\mathcal{E}$  defines which parts of the pattern that the matching should be based on. Finding all records whose values are identical to the corresponding values in the query, corresponds to finding all records  $\mathcal{R}$  such that  $\mathcal{Q}$  connects to  $\mathcal{R}$  through  $\mathcal{I}$ .

The notion of being connected is *directed*. The fact that  $\mathcal{A}$  connects to  $\mathcal{B}$  does not necessarily mean that  $\mathcal{B}$  connects to  $\mathcal{A}$ . For example,  $\mathcal{M}_2$  is connected to  $\mathcal{M}_1$  through  $\{white = white\}$  since

$$\alpha_{\mathcal{M}_2, \mathcal{M}_1} \{mucos-col = mucos-col\} = \{white = white\} \subseteq \mathcal{I},$$

but, as we saw in the example in Sect. 2.4,  $\mathcal{M}_1$  is not connected to  $\mathcal{M}_2$ . Thus, our approach does not exclude the possibility of defining asymmetric similarity measures.

We can generalise Definition 2 by taking the number of non-identical equations in the connecting definition into account. This can be used for defining a natural partial ordering among connecting definitions.

**Definition 3 (Natural Order).** *Let  $\mathcal{A}$ ,  $\mathcal{B}$ ,  $\mathcal{C}$ , and  $\mathcal{D}$  be four definitions. Let  $\mathcal{E} \subseteq \{C = C \mid C \in \text{Com}(\mathcal{A}) \cup \text{Com}(\mathcal{B}) \cup \text{Com}(\mathcal{C}) \cup \text{Com}(\mathcal{D})\}$  and let  $\alpha$  be the method in (1) Further, let  $cd_1 = \alpha_{\mathcal{A}, \mathcal{B}} \mathcal{E}$  and  $cd_2 = \alpha_{\mathcal{C}, \mathcal{D}} \mathcal{E}$ . Then,*

$$cd_1 \prec cd_2 \quad \text{iff} \quad |\{a = b \in cd_1 \mid a \neq b\}| < |\{a = b \in cd_2 \mid a \neq b\}|.$$

Using Definition 3 we get a partial ordering of connecting definitions that is based on what is known *not* to be common among two definitions.<sup>2</sup> This should be compared with many other approaches, e.g., the notion of subsumption of

<sup>2</sup> Other ordering criteria are possible. For example, using set inclusion instead of cardinality would result in a stronger partial ordering.

similitude terms [8,9], where similarity assessment is based on what is known to be common among two structures.

If we return to Example 1, we have  $\alpha_{\mathcal{Q}, \mathcal{M}_1} \mathcal{E} \prec \alpha_{\mathcal{Q}, \mathcal{M}_2} \mathcal{E}$ . Note that we also have  $\alpha_{\mathcal{Q}, \mathcal{M}_1} \mathcal{E} \prec \alpha_{\mathcal{M}_1, \mathcal{Q}} \mathcal{E}$ , which corresponds to the intuition that the variant ( $\mathcal{Q}$ ) is more similar to the prototype ( $\mathcal{M}_1$ ) than vice versa.

If we want, we can use the ordering to compute a numeric similarity value.

Note that Definition 3 is based on the Hamming distance between connecting definitions. By using the complete structure of the connecting definitions we can get an even finer division of the non-identities in the connecting definition:

$$\begin{aligned}\mathcal{NI}_{\mathcal{A}, cd} &= \{a = b \in cd \mid a \neq b \wedge a \in \text{Dom}(\mathcal{A})\} \\ \mathcal{NI}_{\mathcal{B}, cd} &= \{a = b \in cd \mid a \neq b \wedge b \in \text{Dom}(\mathcal{B})\}.\end{aligned}$$

We can use  $\mathcal{NI}_{\mathcal{A}, cd}$  and  $\mathcal{NI}_{\mathcal{B}, cd}$  to define an ordering that is based on what is not defined in  $\mathcal{A}$ , what is not defined in  $\mathcal{B}$ , and what is defined in both, e.g., Tversky's contrast model [14].

### 3.3 More Complex Measures

Our definition of similarity measures is not limited to measures between two definitions.

A possible use of a similarity measure between three definitions is to measure the similarity between two definitions with respect to third definition. This third definition could represent a trend or a prototype, and the similarity between two definitions should be defined as a combination of how similar they are to each other and of how well they follow the trend, or match the prototype. This is similar to the use of a *seed* in [6,7].

Measures with multiple definitions could also be used to capture similarities like “ $a$  is more similar to  $b$  than  $a$  is to  $c$ ” [11].

A special case is when we only have one definition. Our approach makes no distinction between computing and measuring connections between the atoms of two different definitions and computing and measuring connections between the atoms of one single definition. In this sense, our approach is similar to the theory of *high-level perception* [4]. We can use this for defining similarity measures that are based on the comparison of the internal structures of definitions.

*Example 2 (Similarity Based on Internal Structure).* Cases in MedView have an attribute *palpation* which, like *mucos*, is defined in terms of the site of the palpation. Suppose we want to define a similarity measure where examination records in which the two site-values are equal are considered to be more similar than two records in which the two values are not equal.

Let  $\mathcal{M}_1$ ,  $\mathcal{M}_2$ , and  $\mathcal{M}_3$  be the cases in Fig. 1, and let  $\alpha$  be the method in (1). First, we want to compare the two site-attributes:  $\mathcal{E} = \{mucos\text{-}site = palp\text{-}site\}$ . We compute  $cd_1 = \alpha_{\mathcal{M}_1, \mathcal{M}_1} \mathcal{E} = \{l735 = l735\}$ ,  $cd_2 = \alpha_{\mathcal{M}_2, \mathcal{M}_2} \mathcal{E} = \{l122 = l122\}$ , and  $cd_3 = \alpha_{\mathcal{M}_3, \mathcal{M}_3} \mathcal{E} = \{l735 = l737\}$ . Now, we want to measure the

similarity between the three connecting definitions. We define

$$\mathcal{F} \begin{cases} l122 = l122 \\ l735 = l735, \\ l737 = l737 \end{cases}$$

with an identity for every site-value in the co-domain, and

$$\beta_{\mathcal{L}, \mathcal{R}} \begin{cases} \beta = \overline{\mathcal{L}} & \text{if } {}_e\mathcal{E} \in \text{Dom}(\mathcal{L}) \\ \beta = \underline{\mathcal{R}} & \text{if } \mathcal{E}_e \in \text{Dom}(\mathcal{R}) \wedge {}_e\mathcal{E} \notin \text{Dom}(\mathcal{L}) , \\ \beta = \epsilon & \text{otherwise} \end{cases}$$

a two-step variant of (1). If one connecting definition is connected to another using  $\beta$  and  $\mathcal{F}$ , then the two site-values in the corresponding records must be identical. We compute

$$\begin{aligned} \beta_{cd_1, cd_2} \mathcal{F} &= \{l122 = l122, l735 = l735, l737 = l737\} \\ \beta_{cd_1, cd_3} \mathcal{F} &= \{l122 = l122, l735 = l737, l737 = l737\} \\ \beta_{cd_2, cd_3} \mathcal{F} &= \{l122 = l122, l735 = l737, l737 = l737\}. \end{aligned}$$

Thus, the two most similar cases are  $\mathcal{M}_1$  and  $\mathcal{M}_2$ . Note that if we had used one of the basic measures, then  $\mathcal{M}_1$  and  $\mathcal{M}_3$  had been the two most similar cases.

## 4 Estimations and Adaptation

In this section, Definition 2 is extended into a more general estimations process. We also show how estimations can be used for adaptation.

### 4.1 Estimations

Connecting definitions are explicit declarative explanations of why two definitions are similar. This symbolic description can be manipulated by processes that have more domain knowledge about the interpretation of the connecting definition.

If an interpretation of a connecting definition cannot immediately be found, the connecting definition can be used as the starting point for further estimation. The result of this second step can, if necessary, be used as the starting point for a third step, etc. Thus, the computation of a similarity measure is really only the first step in a more general estimation process.

*Example 3.* Cases in MedView contain an attribute *disease*. Suppose we have compared  $\mathcal{M}_1$  and  $\mathcal{M}_2$  in Fig. 1 with respect to their diseases:

$$\mathcal{S} = \alpha_{\mathcal{M}_1, \mathcal{M}_2} \{disease = disease\} = \{Herpes\ labialis = Shingles\},$$

where  $\alpha$  is the method in (1). Without additional domain knowledge we have to conclude that  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are not similar.

Now, suppose we know that *Herpes labialis* and *Shingles* are both viral diseases. This domain knowledge can be defined in a definition  $\mathcal{H}$ :

$$\mathcal{H} \left\{ \begin{array}{l} \textit{Herpetic gingivostomatitis} = \textit{viral} \\ \textit{Herpes labialis} = \textit{viral} \\ \textit{Shingles} = \textit{viral} \cdot \\ \vdots \end{array} \right.$$

We can now define and compute a similarity measure:  $\alpha_{\mathcal{H},\mathcal{H}}\mathcal{S} = \{\textit{viral} = \textit{viral}\}$ .

Thus, using the *estimation*  $\alpha_{\mathcal{H},\mathcal{H}}(\alpha_{\mathcal{M}_1,\mathcal{M}_2}\{\textit{disease} = \textit{disease}\})$  we conclude that  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are, after all, quite similar.

Estimations are a generalisation of Definition 2. Using estimations, we can define the similarity of two definitions as follows: The fewer steps needed in a given estimation process to get a connecting definition that is a subset of  $\mathcal{I}$ , the more similar are the definitions. This can be used to define a natural ordering of definitions, in the same way as Definition 3 define a natural ordering of connecting definitions. This ordering in turn can be used for computing a numeric similarity value.

Apart from being constructed as sequences of applications of similarity measures, estimations may be combined using ordinary logical connectives, forming more complex estimations. Note that, since the connecting definitions are definitions themselves, they can be used in any part of the similarity measure, and not just as a new initial definition as in Example 3.

The distinction between similarity measures and estimations corresponds to the separation of the two tasks of similarity assessment and ranking (preference ordering) [5].

## 4.2 Adaptation

Connecting definitions and estimations are strongly connected to adaptation. The connecting definitions give a precise description of the adaptation needed: equations between non-identical atoms represent what we do not know, connections between atoms which have to be established, or what must be shown to be true. In other words, connecting definitions contain information of which atoms need to be adapted and what they should be adapted to.

If we base the retrieval of cases on the similarity measures in Sect. 3.1, those cases that need the least adaptation, in the above meaning, will be retrieved. This does not necessarily mean that the retrieved cases are adaptable though.

To select cases that are adaptable, some form of adaptation knowledge is needed. In our framework, adaptation knowledge is easily added by defining definitions like the definition  $\mathcal{H}$  in Example 3. Adaptation is performed by the estimation process. In Example 3, the adaptation is trivial, only one extra step is needed, but in general, adaptation may require very complex estimations. Thus, adaptation in the form of an estimation process may be a costly operation, and

the general problem is: How can adaptable cases be retrieved without actually performing the adaptation [13]?

Our solution to this problem is to define special estimations, adaptation methods, like  $\alpha_{\mathcal{H},\mathcal{H}} \mathcal{S}$  in Example 3, and index these estimations according to the adaptation they perform. For example:

$$\mathcal{A} \left\{ \begin{array}{l} (Herpes\ labialis = viral) = \alpha_{\mathcal{H},\mathcal{H}} \{Herpes\ labialis = viral\} \\ (Shingles = viral) = \alpha_{\mathcal{H},\mathcal{H}} \{Shingles = viral\} \\ \vdots \end{array} \right.$$

In general, if a value (an atom)  $v$  can be adapted to a value  $v_a$  using an estimation  $\gamma$ , we need two definitions:

$$\mathcal{A} \left\{ \begin{array}{l} v = v_a \\ \vdots \end{array} \right. \quad \mathcal{A}_m \left\{ \begin{array}{l} (v = v_a) = \gamma \\ \vdots \end{array} \right.$$

The definition  $\mathcal{A}$  is used during case retrieval, in the form of a one-step estimation or by adding it to every case before the computation of the similarity measure is performed. We then use  $\mathcal{A}_m$  as a table: For every non-identity  $v = v_a$  in the connecting definition, if  $v = v_a$  is defined in  $\mathcal{A}_m$ , we know that we can adapt  $v$  to  $v_a$  using the definiens of  $v = v_a$  in  $\mathcal{A}_m$ , i.e., using  $\gamma$ . In this way, new adaptation knowledge is directly available during case retrieval and changes in the adaptation capabilities are immediately reflected in the retrieval step.

The definitions  $\mathcal{A}$  and  $\mathcal{A}_m$  play much the same role as the *adaptation specialists* in [13].

## 5 Discussion

Our framework has been used within the MedView project for interactive exploration of the case base. It has also proved itself useful in the area of functional genomics, where it has been applied to problems connected to the analysis of expression data from proteome analysis of yeast.

The advantages of using a definitional similarity model are simplicity and flexibility. A conceptually simple and uniform model is used: cases and all parts of the similarity measure, including comparison methods, estimations and adaptation knowledge, are given as definitional structures. As a consequence, the output of the model, i.e., the connecting definition, can be used as input in any other part of the model. Like other structured representations, the definitional representation makes it possible to treat subparts of cases as full-fledged cases.

Our similarity model is general enough to capture many different types of similarity measures: ordinal and cardinal measures, asymmetric measures, and measures between any number of objects.

It can be interesting to compare our approach to the feature term approach used in [8,9]. Connecting definitions and similarity measures are definitions themselves, in the same way as similitude terms are feature terms. The retrieval methods of NOOS are similar to the initial definitions in our approach. The initial

definitions enable us to base the similarity assessment on parts of the definitions just as retrieval methods in NOOS prescribe that the comparisons are made only to a subset of relevant feature terms. Adding more equations to the initial definition is a simple example of method refinement in NOOS. Just like similitude terms are declarative descriptions of the similitude among (or structure of) feature terms, connecting definitions are declarative descriptions of connections among definitions. An important difference is that similarity assessment of feature terms is based on what is known to be common among two terms, while similarity assessment of definitions is based on what is known not to be common among two definitions. Another difference is that the method used for similitude assessment of feature term is fixed (subsumption), while the method for similitude assessment of definitions is given by user-definable computation methods.

We use estimations to, step by step, get a clearer understanding of the connecting definition: estimation steps are applied until a “understandable” connecting definition is found. This is similar to the view adopted in [1] where modification rules will be chosen and applied to a given problem and the selected candidate cases, until a common structure of the problem and one candidate case is found.

The definitional representation used in this paper is just a subset of the theory of partial inductive definitions [3]. A more full-fledged calculus would include constructed conditions corresponding to the ordinary logical connectives, including negation and implication. This would make it possible to define similarity measures with more complex initial definitions and computation methods, and to define more complex estimations. For example, non-deterministic definitions, hypothetical reasoning and more.

## References

1. Börner, K.: Structural Similarity as Guidance in Case-Based Design. In: [15] 197–208 391
2. Bunke, H., Messmer, B. T.: Similarity Measures for Structured Representations. In: [15] 106–118 380
3. Hallnäs, L.: Partial Inductive Definitions. *Theoretical Computer Science* **87**(1) (1991) 115–142 381, 391
4. Hofstadter, D. R.: *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, New York (1995) 387
5. Kolodner, J. L.: *Case-Based Reasoning*. Morgan Kaufmann Publishers, Inc. (1993) 380, 389
6. Osborne, H. R., Bridge, D. G.: A Case Base Similarity Framework. In: [12] 309–323 387
7. Osborne, H. R., Bridge, D. G.: Similarity Metrics: A Formal Unification of Cardinal and Non-Cardinal Similarity Measures. In: Leake, D. B., Plaza, E. (eds.): *Case-Based Reasoning Research and Development: Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR-97, Providence, RI, USA, July 1997*. Lecture Notes in Artificial Intelligence, Vol. 1266. Springer-Verlag, Berlin Heidelberg New York (1997) 235–244 387



8. Plaza, E.: Cases as Terms: A Feature Term Approach to the Structured Representation of Cases. In: Veloso, M., Aamodt, A. (eds.): *Case-Based Reasoning Research and Development: Proceedings of the First International Conference, ICCBR-95, Sesimbra, Portugal, October 1995*. Lecture Notes in Artificial Intelligence, Vol. 1010. Springer-Verlag, Berlin Heidelberg New York (1995) 265–276 [380](#), [387](#), [390](#)
9. Plaza, E., de Mántaras, R. L., Armengol, E.: On the Importance of Similitude: An Entropy-Based Assessment. In: [\[12\]](#) 324–338 [380](#), [387](#), [390](#)
10. Ricci, F., Senter, L.: Structured Cases, Trees and Efficient Retrieval. In: Smyth, B., Cunningham, P. (eds.): *Advances in Case-Based Reasoning: Proceedings of the 4th European Workshop, EWCBR-98 Dublin, Ireland, September 23–25 1998*. Lecture Notes in Artificial Intelligence, Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 88–99 [380](#)
11. Richter, M. M., Wess, S.: Similarity, Uncertainty and Case-Based Reasoning in PATDEX. In: Boyer, R. S. (ed.): *Automated Reasoning: Essays in Honor of Noody Bledsoe*. Kluwer (1991) 249–265 [387](#)
12. Smith, I., Faltings, B. (eds.): *Advances in Case-Based Reasoning: Proceedings of the Third European Workshop, EWCBR-95, Lausanne, Switzerland, November 1996*. Lecture Notes in Artificial Intelligence, Vol. 1168. Springer-Verlag, Berlin Heidelberg New York (1996) [391](#), [392](#)
13. Smyth, B., Keane, M. T.: Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval. In: [\[15\]](#) 109–220 [390](#)
14. Tversky, A.: Features of Similarity. *Psychological Review* **84**(4) (1977) 327–352 [387](#)
15. Wess, S., Althoff, K.-D., Richter, M. M. (eds.): *Topics in Case-Based Reasoning. Selected Papers of the First European Workshop, EWCBR-93 Kaiserslauten, Germany, November 1993*. Lecture Notes in Artificial Intelligence, Vol. 837. Springer-Verlag, Berlin Heidelberg New York (1994) [391](#), [392](#)

# Collaborative Maintenance – A Distributed, Interactive Case-Base Maintenance Strategy

Maria Angela Ferrario and Barry Smyth

Smart Media Institute, Department of Computer Science, University College Dublin  
Belfield, Dublin 4, Ireland  
{maria.ferrario,barry.smyth}@ucd.ie

**Abstract.** Case-base maintenance is an important emerging issue for case-base reasoners as they scale-up to handle real-world problems in unstable environments. Overtime the contents of a case-base may become out-of-date or inconsistent with the current target problem space, and maintenance strategies must be devised to help recognise and repair these problems, by deleting, adding, or modifying cases, for example. In this paper we describe a maintenance framework called *collaborative maintenance* (CM), which has been designed to facilitate, monitor, and control the incremental up-date of live, dynamic case-bases. Our approach is novel in that it automatically supports a distributed, interactive maintenance process □ users are permitted to recommend case updates and the collaborative maintenance process ensures that these recommendations are properly reviewed and actioned.

## 1 Introduction

The *maintenance problem* refers to the problem of controlling the evolution of knowledge within a case-based reasoning system so as to optimise some agreed performance metric [11, 12, 18, 20]. This problem has been brought sharply into focus through recent large-scale, real-world case-based reasoning efforts [26]. The problem occurs because many real world domains represent dynamic problem solving environments that are subject to incremental change over time. This has implications for case-based reasoning systems that are designed to operate in such potentially hostile environments. For instance, target problem types may change over time to limit the performance of an existing case-base. Existing problem types may no longer be valid leading to inconsistencies or redundancies within a case-base. Solving the maintenance problem involves developing techniques for monitoring and reacting to changes in performance of various system knowledge sources (the case-base, the similarity knowledge, the adaptation knowledge, etc.). Specifically, maintenance techniques should make it possible to identify shortcomings in system knowledge

sources and to actively bring about the appropriate changes need to repair or optimise these knowledge sources.

To date maintenance research has focused a number of crucial issues such as the control of case-base growth [23, 27], the optimisation of case indices [1], modelling and predicting system performance [22], and the development of a unified maintenance framework [10]. Surprisingly perhaps, relatively little work has been carried out on the initial case collection stage, which is a precursor to many maintenance processes. For example, Borron et al. [4] demonstrate the need for case collection strategies that actively manage the validation of new case knowledge prior to case learning. This is a starting point for the work described in this paper. Specifically, we describe a distributed maintenance strategy, called *collaborative maintenance* (CM) [6, 7], that provides a intelligent framework to support long-term case collection and authoring (Section 4), and that is inspired by recent models of social information filtering, personalisation, and knowledge management [4, 15, 17, 18, 21]. Very briefly, collaborative maintenance supports a distributed maintenance strategy whereby the users themselves have the opportunity to edit case knowledge. In addition, CM manages the review of new case knowledge and provides procedures to govern the acceptance or rejection of new cases. We describe how collaborative maintenance has been implemented in Ulysses, a case-based entertainment planner operating on the Internet (Section 3), but first we begin by surveying related work in the maintenance area.

## 2 Related Work

To date the lion's share of relevant research has focused on the maintenance of the case-base, and we will attend to this work here [11, 12, 18, 20]. However, it must be pointed out that the reach of the maintenance problem extends beyond the case-base to other knowledge sources (containers) such as the similarity knowledge and the adaptation knowledge, and a number of relevant studies have laid the groundwork for looking at the maintenance problem as it relates to these other knowledge sources. For example, a number of studies have investigated techniques for maintaining the similarity knowledge by proposing policies for adjusting the weighting schemes used during similarity assessment in response to changes in the target problem space; Bonzano et al. [3], Munoz-Avila & Hullen [16], and Zhang & Yang [27] describe a number of successful weight-learning policies for example.

Some of the earliest case-base maintenance research dates back the work of Smyth & Keane [22] where the focus is on the development of deletion strategies for removing redundant or harmful cases from a case-base. In particular Smyth & Keane describe a class of competence-guided deletion policies that are effective with case-based reasoning systems. They describe ways of estimating the competence of individual cases and use this information to prioritise low-competence cases for deletion. Recently, a number of alternative deletion addition strategies have also been forwarded. For example, McKenna and Smyth [24] and Zhu & Yang [28] introduce related competence modelling techniques as a means of guiding the addition of cases to an evolving case-base.

Case maintenance strategies can also refine how cases are indexed. For example, Fox and Leake look at how case indices can be incrementally refined through the use of an introspective reasoning technique [8] as a result of planning failures. Alternatively, Aha and Breslow [1] describe an index revision method for conversational case libraries that is guided by external case-library guidelines.

Leake and Wilson [12] have recently focused on maintenance techniques that monitor evolving trends in the performance of a case-based reasoner. For example, they explain how monitoring the relative frequency of reasoning successes and failures can indicate changes in the target problem space, which may need to be reflected by changes in the case-base. Related to this is the issue of identifying case-base "hot spots" which are subsets of cases which have proved to be particularly effective in the past and which may be consulted as a priority during future problem solving sessions; see also [14, 23] for related ideas.

The above maintenance approaches share one common assumption in that they all assume the availability of valid case representations. For example, the case learning and deletion strategies described above all operate over a set of known cases; these cases may have been collected in advance or may be produced as part of the on-going problem-solving process.

In contrast, there has been relatively little research on the core case collection task itself — that is, the task of constructing new valid cases in the first place prior to any pruning or refinement processes. This is a very difficult task to automate or support, and even when a fixed case representation can be assumed there has been relatively little work on how to manage and validate the collection of new cases in order to ensure consistency and accuracy within the case-base. Consistency checking and case validation ultimately requires a strong domain model and this type of knowledge is often unavailable to the CBR system in question — see [18] for related work on case consistency checking. We will look at an example of this in the next section when we outline the Ulysses case-based planner. Ulysses allows users to update its case-base but does not have a domain model which allows these updates to be validated. Collaborative maintenance provides a practical solution by enlisting the support of users as reviewers of any case updates, and then managing the collation and analysis of review results in order to make a decision about whether to accept or reject a given update. In this sense CM is a distributed, interactive case maintenance strategy targeting the case collection and validation tasks.

### 3 ULYSSES – A Case-Based Entertainment Planner

Ulysses is an online, personalised entertainment planner for Dublin city. In its simplest form it provides end-users with information about restaurants, pubs, theatres, clubs, and events in the Dublin area. In addition, it provides a number of advanced functions including: a similarity-based search engine; a case-based entertainment planner capable of generating composite entertainment plans that are personalised for the needs of an individual user; and a home page builder that generates a personalised home page for each user alerting them to relevant upcoming events in their area. In this section we outline the form and function of the Ulysses system.

### 3.1 ULYSSES System Architecture

The Ulysses system architecture is shown in Fig.1; in fact, we illustrate two separate systems □Ulysses and its associated CM component, which will be discussed further in Section 3.

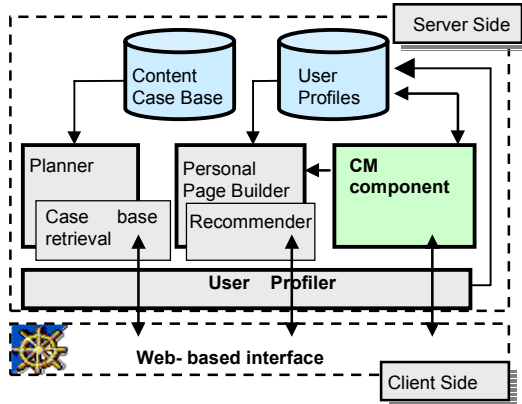


Fig 1. Ulysses □System Architecture

The *content case-base* contains the Ulysses content data including descriptions of restaurants, pubs, clubs, theatre, and events, each described according to a type-specific feature-based schema. For example, restaurants are described in terms of features such as cuisine, price, atmosphere etc. One of the main functions of Ulysses is as an *entertainment planner*. Users can specify their requirements for a night out and the planner will assemble a sequence of events (a meal out, followed by entertainment and a club, for instance) that conform to these requirements. A case-based retrieval engine is used to select individual plan components (restaurants, pubs, events, etc) that are partial matches to the requirements specified [13].

One of the driving motivations behind Ulysses is the production of a personalised environment for like-minded individuals. As such Ulysses uses automatic user profiling techniques to construct detailed profiles on individual users, by tracking their online activity including page requests, click-streams, content rating, and content updates [2, 5, 19]. One of the core features provided by Ulysses is the generation of a personalised home page for each user based on their profiled preferences. Briefly, this page includes information items that are selected by a collaborative filtering recommendation engine. For example, new or up coming relevant events are recommended along with links to the personal home pages of similar users and the virtual community that the user belongs to. The personal home page serves many important functions. Obviously, it enables users to keep up to date with relevant news and events in their sphere of interest, but in addition it helps users to maintain contact with members of their own community and others.

Unfortunately for space reasons it is not possible to discuss Ulysses or its underlying planning and recommendation technologies any further here. However, the interested reader is referred to [6] and to an online demonstration of the system at <http://boffin.ucd.ie/ulysses2000>.

## 3.2 ULYSSES & Collaborative Maintenance

The main focus of this paper is not the Ulysses system per se, but rather the relationship between Ulysses and collaborative maintenance. We have already mentioned that one of the most important problems facing a system like Ulysses is the difficulty in maintaining an up-to-date and accurate content case-base. New content cases must be added, existing cases may need to be revised, and out-of-date cases must be deleted; this is a classic example of the case-base maintenance problem. In particular, the entertainment domain is an example of a very dynamic domain where case-base maintenance is required on a regular and on-going basis.

The main focus of this paper is to explain how this maintenance problem is addressed in Ulysses by using the CM technique. This technique will be discussed in detail in the next section, but for now we would like to set the scene by highlighting that CM has been implemented as an additional layer of technology on top of the Ulysses rather than as an intrinsic part of the Ulysses system. In fact we argue that the CM concept is a general one that can be used for a wide range of distributed maintenance problems and has implications for a wide range of case-base maintenance problems as well as many general knowledge management problems [15].

## 4 Collaborative Maintenance

In Ulysses all registered users are free to add and edit content cases, and these changes are policed by the CM component. The key to collaborative maintenance is that it provides a distributed interactive framework to support the review of changes. It automatically assigns users to the job of reviewing relevant new content cases, selecting reviewers on the basis of their appropriateness and past reliability. Briefly, as users interact with the Ulysses system, and particularly as they add or edit content cases, their progress is closely monitored in order to estimate the standing of a given user as a trusted and reliable source of information [10, 13, 19]. This is a key concept in collaborative maintenance, as a successful review session demands the input of high quality, trusted, and reliable reviewers.

### 4.1 General Approach

The CM process can be divided into the steps shown in Fig. 2. It begins with a new case being added by a user. A set of special users called *information mediators*, which have been found to be reliable and trusted sources of information in the past, are chosen to seed the review process; in a sense these information mediators are the authorities within the user population (see Section 4.4). Individual reviewers are chosen from this set according to their suitability to the current case (see Section 4.5). A review request is then posted to the personal pages of the selected reviewers allowing them to take part in the review process. Once all of the reviews have been submitted or the review period is over) the results are automatically collated and

analysed and a decision is made by the CM system to either reject or accept (as is or with changes) the new content case (Section 4.3); in many ways there are clear similarities between the CM approach and the traditional peer-review model used in academic circles.

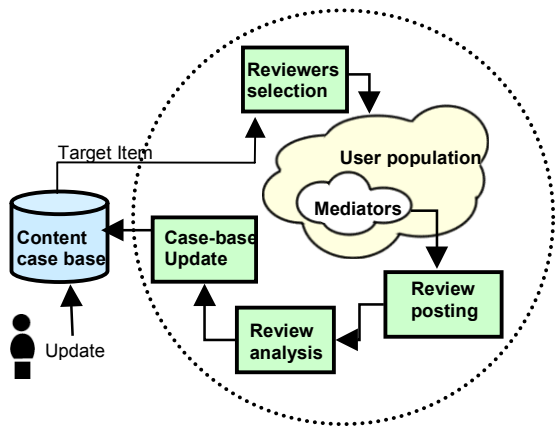


Fig 2. Collaborative Maintenance Cycle

## 4.2 Case Update

Case addition or update is a straightforward process in Ulysses. When a user wants to add or update a content case she fills out a specially prepared HTML form providing values for each of the features associated with the case.

## 4.3 Review Posting and Analysis

Allowing users to freely add and edit cases is fraught with potential problems. Specifically, how can one ensure that user updates are correct or consistent? Collaborative maintenance solves this problem by encouraging users to review all new updates. The key to the success of collaborative maintenance is its ability to select the right users for a given review task. A pool of selected reviewers is chosen for each new or updated content item, and this issue will be discussed further in the next section. Here we will describe the review and verification process itself.

Once suitable reviewers have been selected the content case in question is posted to their personal home page on Ulysses so that they may access the review process the next time they log on. The review process itself is straightforward. Ulysses presents the reviewer with a HTML form containing the case description with form-fields instantiated with the appropriate feature values. The job of the reviewer is to assess whether this feature information is accurate.

For example, Fig. 3(a) shows the review form for a restaurant case (the *target case*) added by some other user (the *contributor*). The reviewer (r) can disagree with the

case as it stands by changing any one of the case features. The result is typically a modified version of the target case (the *reviewed case*) together with a *review score* for this case. The review score is computed according to equation (1) by calculating the similarity between the target case ( $I$ ) and the reviewed case; note that  $r \otimes I$  means the modified version of case  $I$  generated by reviewer  $r$ , that is the reviewed case from  $r$ . For example, Fig 3(b) shows the result of the current review session. Two features were altered (out of a possible 6) resulting in a review score of 78% for this review.

The similarity equation used is shown in (2) and is the standard normalised weighted sum similarity metric which computes the similarity between two cases,  $x$  and  $y$ , by calculating the similarity between corresponding features  $x_i$  and  $y_i$ .

$$ReviewScore(r, I) = similarity(r \otimes I, I) \quad (1)$$

$$similarity(x, y) = \sum_{i=1..n} w_i \cdot sim(x_i, y_i) \quad (2)$$

A typical review cycle will involve the opinions of a number of reviewers, each contributing a potentially modified version of the case and a corresponding review score. At any time during the review process a user can view the partial review results collected so far. These results are presented as feature-value histograms for each the case features. For example, Fig. 3(c) shows the interim results of the review process for the example case.

At the end of the review process the CM system will collate the results and make a decision about the status of the target case. At the present time we use a number of strategies for collating these results and for informing this decision. For example, the simplest strategy is to reject the content item if the mean review score, averaged over all reviewers, falls below a set *rejection threshold*.

However, it is possible for a case to receive a high review score but for the individual reviewers to be in disagreement, in which case the case should not be accepted. This occurs when each reviewer suggests a small number of different changes. For this reason we also compute an *agreement score* between pairs of reviewers,  $r_1$  and  $r_2$ , for content case,  $I$ ; see (3). Essentially this metric computes the similarity between the reviewed cases, using the same similarity metric shown in (2).

$$AgreementScore(r_1, r_2, I) = similarity(r_1 \otimes I, r_2 \otimes I) \quad (3)$$

We then make our reject/accept decision based on a *judgement score*, which is the product of the mean review score and the mean agreement score (4); items that receive a judgement score below a set threshold are rejected.

$$JudgementScore(I, R) = \frac{\sum_{r_i \in R} ReviewScore(r_i, I)}{|R|} \cdot \frac{\sum_{(r_i, r_j) \in R \times R} AgreementScore(r_i, r_j, I)}{|R|^2} \quad (4)$$

Rejected cases are returned to the original contributor for resubmission along with the review results, including agreement scores and suggested reviewer modifications.



An accepted case may still need to be modified according to the suggestions by the reviewers before it is added to the case-base. Currently, CM automatically modifies each case by changing individual features so that they reflect the majority reviewer decision. For example, Fig. 3(d) shows the final results for our example case. Some features have been changed as a result of the review process – the cuisine type and price features have been modified from their original values (*bistro* and *expensive*) to *contemporary* and *moderate*, respectively.

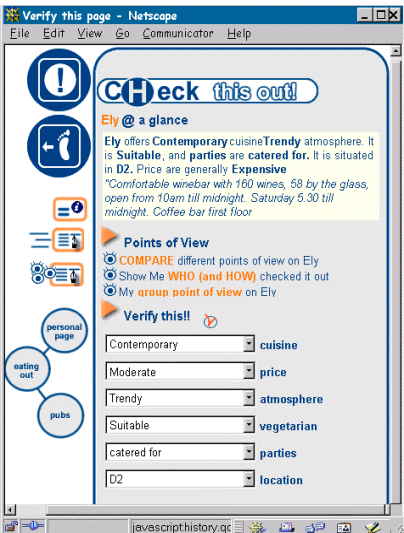


Fig 3a. Initial Reviewer Screen

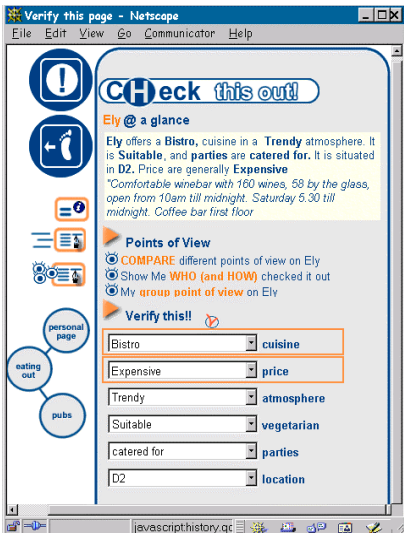


Fig 3b. Suggested Changes

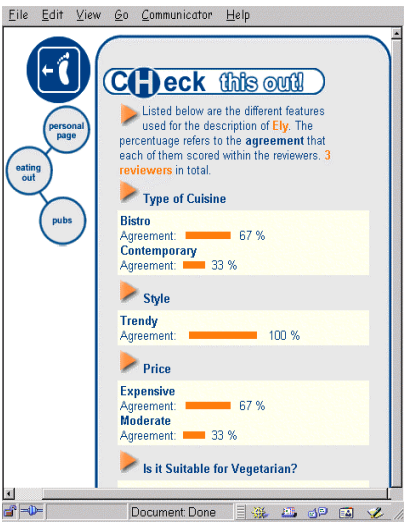


Fig 3c. Review in Progress

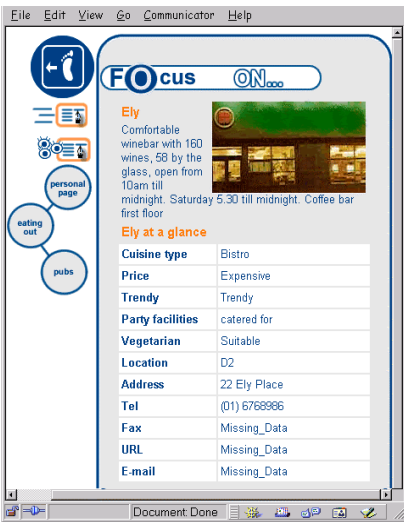


Fig 3d. Accepted Content Item

## 4.4 Information Mediators

In the previous section we described how new or updated cases are assessed by a pool of selected reviewers, and how the results of this process lead to the acceptance or rejection of a case. The success of CM depends critically on the reviewers that are chosen, and there are a number of important trade-offs to consider during this selection process. Reviewers must be knowledgeable about the target case, and they must be trusted and reliable sources of information. Moreover, in many situations reviewers should be selected because they represent a broad cross-section of opinions. It is not wise to select reviewers that are a biased one way or the other with respect to a target case □ this is true for domains (such as the Ulysses domain) where there are no independent criteria to judge case quality, only consensus opinion. On a more practical level, we would also like to select reviewers that are likely to perform their duties, but at the same time ensure that the same reviewers are not continually called upon. We will focus on the reviewer selection process in the next section, but first it is important to understand how CM identifies and classifies trusted and reliable users as potential reviewers.

CM distinguishes various types of users, *information consumers*, *information suppliers*, and *information mediators*. Information consumers and suppliers correspond to the normal Ulysses users that request or supply information items. Information mediators are those users that, over time, have been recognised to be reliable and trusted case contributors. CM targets these users to act as reviewers by scoring users according to their reliability and activity levels within the system.

A user is considered to be a reliable source of information if there is a general agreement, among other users, that the information they provide is correct. In other words, a user is a reliable source of information if their case contributions receive high judgment scores. We calculate the reliability of a user as the *mean value* of all the judgment scores that a user has received for any content items she has added.

Information mediators are users that achieve a high reliability score. During CM the initial reviewer pool contains the  $n$  most reliable users.

## 4.5 Reviewer Selection

The reviewer assignment step involves selecting information mediators that are *relevant* to the target content item. This selection process is important for two reasons. First, obviously there is little to be gained from requesting a review from a user that is not knowledgeable about the target content item; for example, a new night club targeted at 17-25 year olds should not be reviewed by someone who has never before expressed an interest in night clubs. Second, reviewers are more likely to co-operate in the first place if the content item falls within their sphere of interest.

In Ulysses a user profile is made up of information collected both in a passive (implicit feedback) and in an active fashion (explicit feedback). In our system implicit feedback represents the most substantial source of users' information and relies on constantly monitoring users' interactions. Direct feedback, on the other hand, requires explicit participation from the user, as they submit ratings, reviews and comments.

Unfortunately, due to space restrictions it is not possible to discuss these issues in more detail here, but the interested reader is referred to [6] for additional information.

In the previous section we argued that reviewers should be selected because they are trusted and reliable, and because they represent a relevant cross-section of opinions. Collaborative maintenance uses two basic techniques to select reviewers from this information mediator pool. First, a *collaborative* technique is used to select a user ( $u$ ) that is *related* to the original contributor ( $c$ ) — related in the sense that there is a significant overlap between their profile items, which includes the ratings they have provided for previous cases. This is related to the traditional collaborative filtering approach of determining user similarity (see also [9, 13, 19, 25]). The top  $n$  information mediators with the highest profile overlaps are selected as reviewers. Note that, unlike traditional collaborative filtering systems, we are not interested in whether the users are *similar* in the way that they have rated particular cases in the past, because we are interested in selecting a balanced set of reviewers. If we selected reviewers on the basis of rating similarities with the contributor, then we would introduce a bias into the review process, because similar users are likely to agree. As it stands we select users that have looked at some of the same cases as the target user, but that may have differed in their opinions of these cases.

$$Overlap(u, c) = \frac{|u \cap c|}{|u \cup c|} \quad (6)$$

The second selection technique that we use is a *case-based* method. Reviewers are chosen because they have reviewed or rated similar cases to the current target case. Note that this selection is based on the case similarity (2) between the target case and the cases listed in the profile of an information mediator. The similarity between the contributor ( $c$ ) and the potential reviewer ( $u$ ) with respect to the target content item ( $I$ ) is the maximal similarity between the new case (added by the target user) and an item in the mediator's profile. We select the top  $n$  information mediators with the highest similarity to the contributor's target case.

The final result is the selection of a set of reviewers, by combining the selections of the collaborative and content-based approaches, that are appropriate for the new content item and that have proven to be knowledgeable and trusted in the past.

## 5 Conclusions

The maintenance problem is one of the most challenging issues currently facing the CBR community and effective solutions are needed in order to guarantee the success of large-scale CBR systems operating in real-world domains. In this paper we have described a distributed, interactive maintenance technique called collaborative maintenance. Collaborative maintenance makes it possible to allow system users to collaboratively manage the update of case knowledge, by providing a support framework to ensure that updates are reviewed and the appropriate action taken. We have described the mechanics of CM, as it has been implemented in the Ulysses Web-based entertainment planner. A preliminary user-study was carried out earlier this year over a limited selection of users, and the preliminary results are positive in the

sense that our test users found Ulysses to be straightforward to use. Most importantly perhaps, these users did not find the collaborative maintenance process to be obtrusive. However this preliminary study did not provide us with an opportunity to evaluate the long-term use of Ulysses or CM and for this reason our future work will focus on an extended user-study, involving up to 50 users over a substantial length of time. We plan also to investigate automated techniques for the selection of accurate thresholds to be used in the CM cycle (e.g. time limits and number of reviews required for the acceptance/reject of each new added item). Finally we intend to study links between users' behavioural patterns and their motivations to collaborate, in order to better identify the most suitable reward schemes (implicit or explicit) for knowledge-sharing activities.

## References

1. Aha, D. W and Breslow, L. A. (1997) 'Refining Conversational Case Libraries. Proceedings. Int. Conf. Case-Based Reasoning, Providence, RI, USA, pp.267-278.
2. Basu, C., Hirsh C., Cohen, W. (1998). Recommendation As Classification: Using Social and Content-Based Information. Proceedings of the Workshop on Recommendation Systems. AAAI Press, Menlo Park California.
3. Bonzano, A., Cunningham, P., and Smyth, B. (1997) Using Introspective Reasoning to Improve Retrieval in CBR: A Case-Study in Air Traffic Control. Proceedings of the Second International Conference on Case-Based Reasoning. Springer-Verlag, 291-302.
4. Borron, J., Morales, D., and Khlar, P. (1996) Developing and Deploying Knowledge on a Global Scale. Proceedings of the 13<sup>th</sup> National Conference on Artificial Intelligence, Vol. 2. AAAI Press, 1443-1454.
5. Belew, R. K., Hatton, J. (1996) RAVE Reviews: Acquiring Relevance Assessment from Multiple Users. Machine Learning in Information Access. AAAI Spring Symposia.
6. Ferrario, M. A. and Smyth, B. (1999) Collaborative Maintenance. Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science. Cork, Ireland.
7. Ferrario, M. A., Waters, K., Smyth, B. (2000) Collaborative Maintenance in Ulysses. Proceedings of 1<sup>st</sup> International Conference on Adaptive Hypermedia. Trento, Italy.
8. Fox, S. and Leake, D. B. (1995) Using Introspective Reasoning to Refine Indexing. Proceedings of the 14th Int. Joint Conf. on Artificial Intelligence, Montreal, Canada, 1995, pp. 391-397.
9. Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995) Recommending And Evaluating Choices in a Virtual Community of Users. Proceeding of CHI 95, Denver, Colorado, USA.
10. HK, K., Ruström, Å., Waern, A. (1997) Edited Adaptive Hypermedia: Combining Human and Machine Intelligence to Achieve Filtered Information. Proceedings of Flexible Hypertext Workshop, in conjunction with The Eight ACM International Hypertext Conference 97. Southampton, UK.

11. Leake, D. B., and Wilson, D. B. (1998) Categorizing Case-Base Maintenance: Dimensions and Directions. *Proceedings of the 4<sup>th</sup> European Workshop on Case-Based Reasoning*, Springer-Verlag, 196-207.
12. Leake, D. B., and Wilson, D. B. (1999) When Experience is Wrong: Examining CBR for Changing Tasks and Environments. *Proceedings of the 3<sup>rd</sup> International Conference on Case-Based Reasoning*, Springer-Verlag, 218-233.
13. Maltz, D. and Ehrlich, K. (1995) Pointing The Way: Active Collaborative Filtering. *Proceedings of CHI 95*, Denver, Colorado, USA.
14. McKenna, E. and Smyth, B. (In Press) Visualising the Competence of Case-Based Reasoners. *Journal of Applied Intelligence: Special Issue on Interactive Case-Based Reasoning*.
15. Meltsner, K. J. (1997) Opportunities for AI applications in Knowledge Management, AAAI Spring Symposium Artificial Intelligence in Knowledge Management Stanford University, USA.
16. Munoz-Avila, H., Hullén, J. (1996) Feature Weighting by Explaining Case-Based Planning Episodes. *Proceedings of the 3<sup>rd</sup> European Workshop on Case-Based Reasoning*. Springer-Verlag, 280-294.
17. Perkowitz, M., and Etzioni, O. (1998) Toward Adaptive Web Sites: Conceptual Framework and Case Study. *Proceedings for Second Workshop on Adaptive Hypertext and Hypermedia in Hypertext'98*. Pittsburgh, USA.
18. Racine, K. and Yang, Q. (1997) Maintaining Unstructured Case Bases. *Case-Based Reasoning Research and Development. Proceedings of the 2<sup>nd</sup> Int. Conference on Case-Based Reasoning*, Providence, RI, USA, pp. 553 □564.
19. Shardanand, U. and Maes, P. (1995) Social Information Filtering: Algorithms for Automating "Word of Mouth". *Proceeding of CHI 95*, Denver, Colorado, USA.
20. Smyth, B. (1998) Case-Based Maintenance. *Proceedings of the 11<sup>th</sup> Int. Conference on Industrial and Engineering Applications of AI and Expert Systems*, Castellón, Spain.
21. Smyth, B. & Cotter, P. (1999) Surfing the Digital Wave: Generating Personalised Television Guides using Collaborative, Case-based Recommendation. *Proceedings of the 3<sup>rd</sup> International Conference on Case-based Reasoning*. Munich, Germany.
22. Smyth, B. & Keane, M. T.: Remembering to Forget: A Competence Preserving Deletion Policy for Case-Based Reasoning Systems. *Proceedings of the 14<sup>th</sup> International Joint Conference on Artificial Intelligence*. Morgan-Kaufmann. (1995) 377-382
23. Smyth, B. and McKenna, E. (1998) Modelling the Competence of Case-Bases. *Proceedings of the 4<sup>th</sup> European Workshop on Case-Based Reasoning*, Springer-Verlag, 208-220.
24. Smyth, B. and McKenna, E. (1999) Building Compact Competent Case-Bases. *Proceedings of the 3<sup>rd</sup> International Conference on Case-based Reasoning*. Munich, Germany, pp. 329 - 342.
25. Ungar, L. H. and Foster, D. P. (1998) Clustering Methods For Collaborative Filtering. *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park California.
26. Watson, I. (1997) Applying Case-Based Reasoning: Techniques for Enterprise Systems, Morgan-Kaufmann.

27. Zhang, Z., Yang, Q. (1999) Dynamic Refinement of Feature-Weights in CBR Using Quantitative Introspective Learning. Proceedings of the International Joint Conference in Artificial Intelligence 1999.
28. Zhu, J. and Yang, Q. (1999) Remembering to Add: Competence Preserving Case-Addition Policies for Case-Base Maintenance. Proceedings of the 16<sup>th</sup> International Joint Conference on Artificial Intelligence, Stockholm, Sweden, pp. 234-239.

# A Unified CBR Architecture for Robot Navigation

Susan Eileen Fox

Mathematics/Computer Science Department, Macalester College  
1600 Grand Avenue, Saint Paul, MN 55105  
`fox@macalester.edu`  
`http://www.macalester.edu/~fox`

**Abstract.** A delivery robot navigating in the real world and contending with multiple goals and priorities benefits from combining deliberative and reactive planning. Deliberative planners anticipate and optimize actions, and manage multiple goals at once. Reactive planners respond flexibly and moment-to-moment to a changing world, based on incomplete and flawed knowledge. This project proposes using case-based reasoning to fully integrate high-level and low-level planning techniques, along with other reasoning tasks of the system. The resulting robot allows case selection to mediate between methods. In addition, the robot learns by recording new reactive behaviors, storing new plans it creates, and introspective learning of retrieval features.

## 1 Introduction

A message delivery robot navigating in the real world requires a range of skills, from avoiding obstacles and moving in a set direction to making plans and juggling sets of goals and priorities. The robot must be able to respond quickly to changes in the world, to assess and verify its location, to keep track of deadlines, and to compensate for missing or faulty information about the world. Its control system should be flexible and responsive both in its approach to its goals and to its current actions. Neither deliberative planning methods nor reactive planning methods capture the entire range of tasks for such a robot.

A deliberative planner constructs a complete plan before beginning execution, and usually makes simplifying assumptions: i.e. that the world doesn't change while the plan is being constructed. A deliberative planner may anticipate problems, construct contingency plans, and take into account a range of goals and priorities in designing its plan. A reactive planner, by contrast, selects a set of low-level behaviors by examining its current situation, making a quick decision about immediate actions to take, and re-assessing its choice every few moments. Reactive planners can easily implement obstacle avoidance, wall following, and dynamic adjustment of the robot's affective parameters. It is more difficult for a reactive planner to mediate among a set of (possibly) competing goals, or to take a high-level, long-term view of its actions.

In the past, deliberative and reactive planning were seen as competing philosophies for controlling a robot. More recently, attempts have been made to integrate the approaches in order to create a system that has the best features of both. The strengths of reactive planning are complementary to the strengths of deliberative planning.

Many systems which integrate deliberative and reactive planning use separate reasoning methods to perform each kind of planning task. Having separate methods increases the complexity of the overall system, and means that the system must explicitly choose when to apply high-level reasoning and when to apply a low-level behavior.

This project, RUPART,<sup>1</sup> integrates deliberative and reactive planning using a single mechanism: case-based reasoning. Both reactive behaviors and high-level plans may be encoded as cases, and retrieved using the same indices and similarity assessment. The resulting system automatically picks the best level of attention based on its current knowledge, its current goals, and its current situation in the world.

Using a unified, *learning* architecture means that RUPART improves by learning from its experience at all levels. RUPART learns by storing the new high-level plans it creates, by recording the new reactive behaviors it performs, and by introspective learning of similarity assessment parameters.

RUPART uses case-based reasoning to implement all of its reasoning needs, including parts of the CBR process itself. It relates to, and parallels, the RILS system which also re-uses the CBR process and which provides to RUPART case-based introspective learning [5].

RUPART operates on a Pioneer II DX robot which has a gripper, sonar, and some vision capabilities. The robot navigates in the unmodified hallways of the Olin-Rice Science Center at Macalester College.

In this paper I provide an overview of the RUPART system and the issues that arise in its task and domain. I discuss the issue of robot navigation and compare deliberative and reactive planning. I describe the task and domain of RUPART, its architecture, and discuss how similarity assessment mediates between deliberative and reactive planning.

## 2 Robot Navigation: A Rich, Complex Domain

RUPART is a delivery robot navigating through the hallways of an office building, carrying objects or messages from place to place. New requests for its services may come in at any time, and it may be asked to visit locations which it has never seen before. This kind of task requires both high-level reasoning and flexible responses to a complex physical environment.

---

<sup>1</sup> Robot Uniting Plan-Ahead and Reactive Tasks



## 2.1 Complexity of Low-Level Navigation

Historically, most planning methods in AI were “deliberative:” the planner attempted to create a complete plan which could be executed without much further effort by the robot. These methods worked well on complex, abstract domains. Applied to a robot in the real, messy, inexact world, however, these systems failed (e.g., [11]).

A physical robot faces an extremely complex, rich domain. The robot cannot have a complete internal representation of the world, the world changes in a multitude of ways while the robot is planning or acting, and the robot’s interface to the world (sensors and effectors) is laden with uncertainty and inaccuracy.

Because of the difficulty of handling execution issues with a deliberative planner, research on navigation of mobile robots has shifted in recent years to “reactive planning.” Reactive planners emphasize low-level behaviors, a tight coupling between sensing and acting, and quick decisions based on each moment’s situation [2]. Brooks’ “subsumption architecture” deliberately avoids the creation of an explicit world model at all. Systems taking this “bottom-up” approach to robot control are able to exhibit a number of fairly sophisticated behaviors: wall-following, obstacle avoidance, exploration, object location, etc.

The SINS system demonstrates reactive planning using a case-based framework [14]. In SINS, stored cases represent parameters for controlling and changing reactive behaviors over a window of time. The system uses multi-strategy learning to develop a set of good cases for its overall behavior, and does no high-level planning.

Connectionist networks have recently been used for reactive robot control. Meeden trained a robot using reinforcement learning to exhibit behaviors like approaching or avoiding a light source [9].

Several researchers also use reactive planning to control software agents in a simulated domain. Pengo, a system which plays a video game, demonstrates the power of reactive planning to succeed in a rapidly changing domain, with an agent with fixed long-term goals and little need to plan strategically [1]. The TRUCKER system uses “Reactive Action Packets,” a somewhat higher level description of reactive behaviors, to simulate a delivery truck [3].

A reactive planner’s strengths are in its flexible, fast response to its immediate situation. It is more difficult for a reactive planner to prioritize a set of goals it may have, or to anticipate and avoid pitfalls based on an internal map of its domain.

## 2.2 Complexity of Goal Management

A robot whose task is to explore an unknown area may depend on reactive planning techniques while requiring little high-level reasoning. However, high-level reasoning is more important for a “messenger robot” whose task requires it to juggle multiple requests in a timely fashion. The robot may have deadlines for the completion of a given task, and must be able to identify and navigate to a wide range of specific locations. TRUCKER is the only example given

above which manages multiple goals at once: it handles its goals based on fixed priorities and opportunistic goal achievement.

The strength of a deliberative planner is its ability to combine the system's overall goals and to plan on the basis of a high-level view of the domain. Deliberative planners (those based on the STRIPS language [15], among others) have limitations when it comes to plan execution, but they do have the advantage of viewing their tasks at the domain level, and can therefore anticipate and avoid problems further into the future. When a robot's performance must be optimized in order to meet its goals, high-level reasoning can provide important guidance for controlling lower-level navigation.

Planners which generate a plan from scratch every time are often less efficient than planners which re-use old solutions where possible. Case-based planners provide a performance advantage through the re-use of old solutions. ROUTER combines a traditional route planner with a case-based one [7], for generating directions for traversing a college campus. NICOLE uses asynchronous plan retrieval to permit reasoning to occur in parallel with retrieval; it focuses on a multi-strategy approach to merging planning experiences [13].

### 2.3 Combining Reactive and Deliberative Planning

Many robot navigation systems in recent years have combined deliberative with reactive techniques. Most such systems separate reactive from deliberative behaviors, and explicitly mediate between them. Many, also, incorporate no learning from experience.

As one example, Trulla is an optimal path planner which uses a "potential field" method for indicating the direction, weight, and related subgoals at each location on its map. Its reactive system uses the information to avoid less desirable paths, and for recovering from unexpected obstacles [10]. It uses completely separate techniques for path planning and reactive planning, and does not learn.

The ROBBIE system, a predecessor to RUPART, used "planlets" similar to Firby's RAP model for reactive execution of high-level plans generated by a case-based planner [6]. The high-level plan steps specified subgoals for the low-level system. ROBBIE used CBR for both high-level planning and selection of reactive planlets. However, the planlets were still at a relatively high level of abstraction, for a simulated robot in a simulated domain. ROBBIE learned new high-level plans, but could not learn low-level planlets.

ROGUE is a case-based planner for a physical robot that does combine high-level and reactive plan techniques [8]. ROGUE's task domain is similar to RUPART's, and ROGUE does learn from experience. However, ROGUE uses separate mechanisms for its reactive behaviors and its path and task planning.

## 3 Re-using Case-Based Reasoning

A system which re-uses CBR for all of its reasoning needs has a more simple structure than one which uses multiple reasoning methods. Besides being intellectually more pleasing, a streamlined structure enables the easy construction of

a model for introspective learning, which allows RUPART to alter its indexing features and weights in response to retrieval failures. Re-using the case-based reasoning process expands the parts of the system at which learning from experience may occur.

The IULIAN system performs discovery learning, using case-based reasoning processes for retrieving old solutions, for constructing experiments, for constructing theories, and for introspecting about its solution process [12].

ROBBIE and RILS, from which RUPART descends, re-use CBR for portions of the case-based planning process: for selecting plans, for selecting index modification rules, for adaptation strategy selection, and for reactive planlet selection [6,5]. RILS, in addition, uses case-based reasoning to implement its introspective learning module. Some internal uses of the CBR process use case indices which are essentially orthogonal to the plan indices. RUPART, however, uses exactly the same indices for deliberative and reactive plans. RUPART uses the introspective-learning module from RILS, although its importance to RUPART will not be seen until further experimentation is complete.

## 4 The Robot Messenger Task and Domain

RUPART's task is to navigate the second floor of the Olin-Rice Science Center at Macalester College, going from location to location carrying messages, packages, or letters. Locations include offices, classrooms, laboratories, hallways, or a large atrium area. There are several pillars and other permanent obstacles in the atrium area, plus several balconies overlooking the first floor. Except where stairways connect to the balconies, a railing assists the robot to identify the boundaries of the danger zones. Figure 1 shows an abstract layout of the robot's domain.

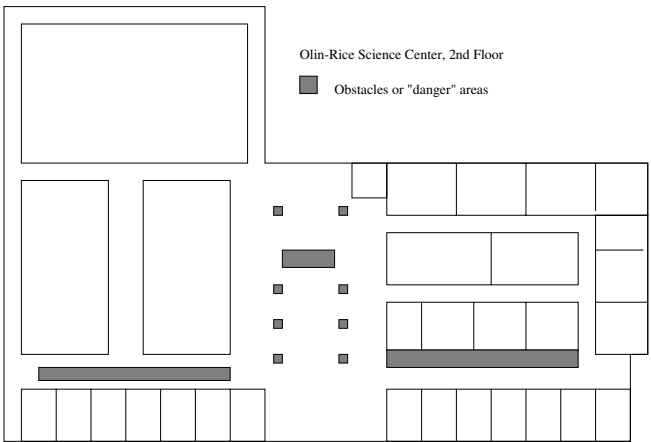


**Fig. 1.** An abstract floor-plan of RUPART's physical domain

The atrium area is frequently filled with obstacles such as tables and chairs in a variety of configurations. As this is a real building with real uses, other temporary obstacles appear frequently and unpredictably. Further, RUPART must share the space with human users.

Locations are specified to RUPART by overlaying a grid of approximately 50 cm by 50 cm over the map given above, and describing locations by coordinates in that grid. Certain locations are defined for RUPART by name as well: i.e. offices, classrooms, and labs. The names relate to a particular coordinate taken to represent that space.

Tasks are given to RUPART asynchronously, and may include time limits and priority values. For example: RUPART could be asked to pick up a printout in the computer lab and take it to classroom 243 before 1 pm, or to take a message to Office 206 with highest urgency.

The nature of this domain challenges RUPART to perform at a high level both in moment-to-moment navigation and in planning which goals to perform and which routes to take. RUPART *must* integrate both low and high level reasoning abilities.

## 5 The Architecture of RUPART

Figure 2 illustrates RUPART's basic architecture. The details of index creation, plan creation, adaptation, and introspective reasoning are omitted for simplicity in this figure. Most reasoning is done using the same case-based reasoning process, including index creation and adaptation within the case-based planning process itself [4,5]. Besides the central case memory containing all the varied cases of the system, there is a global map which all portions of the system access. The map is updated to reflect the observed world as the robot progresses.

RUPART cycles rapidly through a decide-implement-and-act loop during which it assesses its current situation and decides whether it needs to select a new action to take. RUPART chooses to re-plan when its situation varies significantly from either its expectations or the situation when re-planning last occurred. RUPART considers its current set of goals, priorities, and deadlines, as well as its sensory input, estimated location, and estimates of certainty about its location and its current actions.

If it chooses to update its behavior, it constructs an index describing its current situation and retrieves a small number of cases which best match its situation. It uses a modified nearest neighbor heuristic for similarity assessment. The cases retrieved may be "meta-plans" which suggest strategies without specifics (i.e., "construct a plan from scratch" or "wander in the right general direction"), or "plan steps," high level plan descriptions which may need to be adapted to fit the current situation, or reactive behaviors which fit the current situation. Based on what is retrieved, RUPART may create or adapt a high-level plan, or execute a low-level behavior. Typically, while these processes are being carried out it also pursues some reactive behavior in parallel.

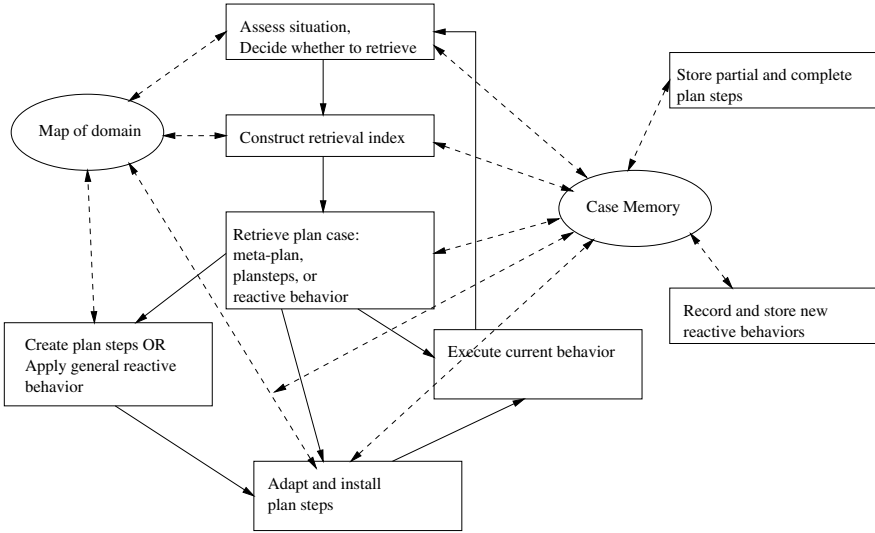


**Fig. 2.** The RUPART Architecture

Separate processes handle the learning components of RUPART. The plan storer packages and stores high-level plan steps from RUPART's immediate history whenever re-planning alters the plan steps to be considered. The behavior recorder similarly examines RUPART's immediate history whenever significant changes are made to the reactive behaviors being executed, and packages the behavior for future use. This mechanism is conceptually similar to that used by SINS to create reactive behavior cases [14].

## 5.1 Situation Assessment

Situation assessment requires two main tasks. The first is to determine how consistent RUPART's sensory inputs are to its predicted location and to the location used when last re-planning. The second is to manage the changing goals and priorities RUPART is faced with. The assessor also updates the global map to reflect the new data RUPART has collected about its world.

For the first task, RUPART uses a simple distance heuristic to determine whether the features its sensors are detecting correspond to its current map. RUPART gets sonar data about the world around it; in the near future I anticipate adding some simple visual input as well. If the difference between predicted and actual data exceeds a pre-set bound, then RUPART chooses to retrieve a new plan of action no matter what else might occur. Similarly, RUPART re-plans when the current sensor data differs significantly from the sensor values at last re-planning.

Managing multiple goals and deadlines is a more complex task. RUPART orders the system's goals based on (1) the priority of the goal (goals with im-

pending deadlines get highest priority) and (2) the proximity of the goal location to RUPART's current estimated location. At each pass, the situation assessor considers whether the order of goals has changed. If it has, then the new top goal is used as part of the situation description for retrieving a new plan of action.

## 5.2 RUPART's Planning

The interaction between deliberative and reactive planning is mediated by case-based indexing and retrieval. Based on features relevant to both high-level and low-level planning actions, RUPART retrieves plan cases which are the best matches: high-level plans, reactive behaviors, and meta-plans. It "chooses" to deliberate or to react based on which retrieved case is the best match.

A high-level plan describe steps for the robot; each step becomes a subgoal for the reactive component. Figure 3 shows a portion of a typical high-level plan for RUPART which takes it from the atrium to a classroom on the north side of the building. RUPART's case memory initially includes a small number (less than 10) of hand-coded plans, augmented by as new plans are generated and executed.

```
( ...
  (move (dir north)
        (in (location-name atrium))
        (to (obstacle-type wall)))
  (move (dir east)
        (in (location-name atrium))
        (to (location-name north-olin-hallway)))
  (move (dir east)
        (in (location-name north-olin-hallway))
        (to (location-name olri-245)))
  ...)
```

**Fig. 3.** A plan description for moving from the atrium to a classroom

Reactive behaviors include parameter values for speed, direction, obstacle avoidance, and so forth. Behaviors are indexed by the situations in which the behavior is appropriate. Figure 4 shows a typical reactive behavior for navigating a fairly open area where pedestrian traffic is common. RUPART begins with just reactive meta-plans and learns new behaviors as it goes.

RUPART's memory also includes meta-plans which act as fall-back retrievals. Meta-plans have indices that match weakly a wide range of specific situations. The system will automatically select a meta-plan when no specific case (deliberative or reactive) matches the current situation above some threshold. Meta-plans may encode generic reactive behaviors, such as "wander at random" or "find and follow a wall," or may suggest alternative methods for generating high-level plans

```
( ...
  (speed moderate)
  (acceleration slow)
  (orient-by direction)
  (obst-distance maximal)
  (obst-response pause)
  ...)
```

**Fig. 4.** A reactive behavior for an open area with many pedestrians: moderate speed, keep away from obstacles, but pause in case they move away

(e.g., throwing out the current plan steps and starting over from the current point).

The following scenario illustrates how RUPART's single method of case retrieval would choose deliberative or reactive planning when faced with the unexpected.

Suppose RUPART needs to get to a location which requires passage down a hallway that is unexpectedly blocked by boxes. When RUPART sensors detects the blockage, its actual situation does not match its expectations, so it re-plans by retrieving cases for the new situation. RUPART might retrieve a reactive behavior for navigating around partial blockages. If no such behavior exists, or the blockage proves impassable, RUPART might fall back on a meta-plan which suggests discarding the old high-level plan and re-planning from scratch. This second round of retrievals, taking into account the blockage in the hallway, might retrieve a high-level plan that takes an alternate route or, if none exists, a reactive behavior to explore away from the obstacle.

### 5.3 The Global Map

RUPART's map is a central repository for its knowledge about the world. At the moment, it is the one piece of knowledge that is external to the case memory, for the convenience of high-level plan creation and plan adaptation.

RUPART is given a basic map to start with, showing the permanent structures of its world (walls, doorways, etc.) RUPART updates the map as it travels, using discrepancies between the internal map and the world as learning opportunities. A discrepancy could occur because (1) a transient obstacle is in the way, (2) the map is incorrect, (3) its sensors are incorrect, or (4) RUPART is not where it thinks it is.

To address these possibilities, RUPART marks objects in the map with a measure of the object's "permanency." When it detects a significant inconsistency between the map and its sensors (as defined by the distance heuristic discussed above) it adds the sensor data to the map at the lowest level of permanency. As the object persists in its sensors, its permanency level is increased. If the object is not reinforced by additional encounters with the robot, its permanency level tends to decay. Objects that no longer appear to the robot's sensors will

eventually decay off the map. Only those feature coded as permanent features of the map cannot be removed.

This method allows RUPART to update its map to include data from its sensors, and to plan based on that data, such that the map automatically recovers from the occurrence of movable objects, sensor flukes, or location confusions.

## 5.4 Learning in RUPART

At this point in time RUPART is still a work in progress, and the learning component of RUPART has not been the focus. Methods are in place, but untested, to learn by recording reactive behaviors, by storing higher-level plans, and by introspectively altering the indexing of high level plans, if not reactive ones.

Currently, RUPART records the high-level plan steps which are successfully used to guide reactive planning. When the plan is complete or is discarded in progress, RUPART packages the plan steps executed up to that point as a plan and adds it to memory.

It similarly records the reactive parameters used by the system between one re-planning event and the next and stores them under the index describing the situation in which they started. Currently, RUPART avoids storing identical cases, but the problem of avoiding overpopulation of reactive cases is still under consideration.

RUPART incorporates the introspective learning component derived from the RILS system. This component focuses on changes to the indexing criteria based on retrieval failures: adding indexing features or altering the weights used for similarity assessment. RUPART does not depend on this learning, and it has not been fully tested.

## 6 Similarity Assessment of Multi-leveled Plans

RUPART uses its similarity assessment method to mediate between planning and action. Therefore, a good measure of similarity is absolutely critical to the performance of the overall system. The key difficulty is the difference in scale: reactive behaviors describe a momentary situation, whereas high-level plans describe actions over the long term. To combine both scales, the indices for plan cases include information that ranges in detail from a snapshot of the sensor data (from the last few time steps) to the goal which is currently preminent. Table 1 lists the features used to index plan cases, along with how distance is measured for each.

Similarity assessment uses a modified nearest-neighbor heuristic, where distance is calculated as the weighted sum of the distances between each feature. Numeric values such as urgency and confidence values use the scalar difference; vectors sum the differences at each value. Non-scalar data features are unified, and the distance estimated from the level of match. Each  $D_i$  value is combined using the following formula, where  $i$  ranges over each indexing feature:



**Table 1.** Similarity assessment: weighted sum of distances for each feature calculated as shown below

| Index feature                              | $D_i$                                                |
|--------------------------------------------|------------------------------------------------------|
| Snapshot of sensor data                    | $\sum_j  f_{jcurrent} - f_{jindex} $                 |
| Estimated location on the map $(x_e, y_e)$ | $\sqrt{(x - x_e)^2 + (y - y_e)^2}$                   |
| Preeminent goal                            | $Unify(G_{current}, G_{index})$ , in range $[0, 10]$ |
| Plan in effect?                            | 1 if yes, 0 if no                                    |
| Current plan step if any                   | $Unify(P_{current}, P_{index})$ , in range $[0, 10]$ |
| Urgency $U$                                | $ U_{current} - U_{index} $                          |
| Location confidence $LC$                   | $ LC_{current} - LC_{index} $                        |
| Action confidence $AC$                     | $ AC_{current} - AC_{index} $                        |

$$D = \sum_i w_i \times D_i$$

Once  $D$ , the distance measure is calculated, RUPART retrieves those cases within some small distance of the best case, ordered best to worst.

## 7 Conclusions

This project is still in its early stages. The general planning and learning frameworks are in place, experimentation with the actual robot is in progress, and much tweaking of parameters remains to be done. The learning of RUPART is rudimentary at this point, as the focus is on successfully retrieving and using cases hand-coded for its memory.

Ultimately, RUPART will start out with a fairly impoverished case library, and will learn both reactive behaviors and plans that are specific to its particular domain.

Nonetheless, RUPART demonstrates already the ability of case-based processes to be re-used for a multitude of reasoning tasks within a single system. Beyond that, it demonstrates the power of combining reactive and deliberative planning within a single framework. RUPART's behavior is driven, not by hard-coded heuristics for when to plan ahead and when to take action, but by the system's immediate goals, situation, and the knowledge in its case library.

Once learning in RUPART has been improved, RUPART should present a powerful package for robot navigation under time and goal constraints. It will combine reactive and deliberative planning, switching dynamically from one to the other as its needs dictate. It will improve with experience by learning new

reactive behaviors and new high-level plan cases, plus introspective learning to tweak its case-based mechanisms themselves. It will demonstrate flexibility in the face of a complex domain, in maneuvering physically in the world, in juggling multiple goals, and in learning from experience to perform well in its domain.

## References

1. P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth Annual National Conference on Artificial Intelligence*, Seattle, WA, July 1987. AAAI. 408
2. R. Brooks. Intelligence without representation. In *Proceedings of the Workshop on the Foundations of Artificial Intelligence*, Cambridge, MA, 1987. MIT. 408
3. R. J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, Computer Science Department, 1989. Technical Report 672. 408
4. S. Fox. *Introspective Reasoning for Case-Based Planning*. PhD thesis, Indiana University, Computer Science Department, 1995. IUCS: Technical Report 462. 411
5. S. Fox. A new model of reflective introspective learning. In *Proceedings of the Eleventh International Florida Artificial Intelligence Research Symposium Conference*, 1998. 407, 410, 411
6. S. Fox and D. Leake. Learning to refine indexing by introspective reasoning. In *Proceedings of the First International Conference on Case-Based Reasoning*, Sesimbra, Portugal, October 1995. 409, 410
7. A. Goel, T. Callantine, M. Shankar, and B. Chandrasekaran. Representation, organization, and use of topographic models of physical spaces for route planning. In *Proceedings of the Seventh IEEE Conference on AI Applications*, pages 308–314. IEEE Computer Society Press, 1991. 409
8. K. Haigh and M. Veloso. Planning, execution and learning in a robotic agent. In *The Fourth International Conference on Artificial Intelligence Planning Systems 1998 (AIPS'98)*, pages 120–127, Pittsburgh, Pennsylvania, June 1998. 409
9. L. Meeden. *Towards Planning: Incremental Investigations into Adaptive Robot Control*. PhD thesis, Indiana University, Computer Science Department, 1994. 408
10. R. Murphy, K. Hughes, and E. Noll. An explicit path planner to facilitate reactive control and terrian preferences. In *1996 IEEE International Conference on Robotics and Automation*, volume 3, pages 2067–2072, Minneapolis, MN, April 1996. 409
11. N. J. Nilsson. Shakey the robot. Technical Report 323, AI Center, SRI International, Menlo Park, CA, 1984. 408
12. R. Oehlmann and P. Edwards. Introspection planning: Representing metacognitive experience. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*, Stanford, CA, March 1995. AAAI. In press. 410
13. A. Ram and A. Francis. *Multi-plan Retrieval and Adaptation in an Experience-based Agent*, chapter 10. AAAI Press, Menlo Park, CA, 1996. 409
14. A. Ram and J. Santamaria. Multistrategy learning in reactive control systems for autonomous robotic navigation. *Informatica*, 17(4):347–369, 1993. 408, 412
15. E. D. Sacerdoti. A structure for plans and behavior. Technical Report 109, SRI Artificial Intelligence Center, 1975. 409

# Maintenance of a Case-Base for the Retrieval of Rotationally Symmetric Shapes for the Design of Metal Castings

Tony Mileman, Brian Knight, Miltos Petridis, Keith Preddy, and Patrick Mejasson

University of Greenwich, School of Computing and Mathematics,  
Park Row, Greenwich, London SE10 9LS UK  
{b.knight,m.petridis}@greenwich.ac.uk

**Abstract:** In this paper, we discuss the problem of maintenance of a CBR system for retrieval of rotationally symmetric shapes. The special feature of this system is that similarity is derived primarily from graph matching algorithms. The special problem of such a system is that it does not operate on search indices that may be derived from single cases and then used for visualisation and principle component analyses. Rather, the system is built on a similarity metric defined directly over pairs of cases. The problems of efficiency, consistency, redundancy, completeness and correctness are discussed for such a system. Performance measures for the CBR system are given, and the results for trials of the system are presented. The competence of the current case-base is discussed, with reference to a representation of cases as points in an  $n$ -dimensional feature space, and a Gramian visualisation. A refinement of the case base is performed as a result of the competence analysis and the performance of the case-base before and after refinement is compared.

## 1 Introduction

This paper addresses the problem of retrieval of rotationally symmetric shapes from a case base designed as a design assistant to be used in the metal casting industry. In a previous paper [7] a support tool was described, based on a traditional componentisation of rotationally symmetric shapes, with components of known cooling modulus. This paper describes further research into the use of this traditional componentisation as a shape representation for retrieval. The work concentrates on the connectivity of graphical components. There are of course other attributes which a practical case-based system can exploit, such as scale of cast objects and components, textual description, and traditional classifications of objects to be cast. There are plans to include these search indices to enhance the performance of the system described here.

Out of a range of solids processing methods for the mass production of components; for example, casting, forging and machining, casting is the generally the cheapest. However, the problem with casting is one of quality, which depends on the existence of casting design knowledge. The advantages of a CBR system, capable of containing detailed information on the design process for products, devolve from its ability to realise casting know-how as a valuable asset. The knowledge of how to cast a product soundly within tight cost constraints is the result of a huge investment on the part of industries, universities and government over many years. Although the value of design knowledge is widely recognised throughout the industry, the management of design knowledge is often ad hoc in some respects. Design histories are often lost, or banished to paper files that are difficult to search. Also, design engineers retire [27], or move away leaving inadequate design records.

There are many problems faced by a casting design engineer, centering on the physical freezing processes. Foremost among these is shrinkage in the mould, which can give rise to porosity and areas of structural weakness [1]. Other practical problems arise during pattern making and subsequent machining of the cast part. Many software tools have been developed to assist the designer. In a recent survey, Jolly [2] found that the foundry industry is looking for software that can not only predict problems that occur during metal solidification (such as shrinkage porosity) but also, having predicted these problems, to propose intelligent solutions to problems found. Current commercial casting software can be classified into two broad areas: intelligent knowledge-based systems (IKBS), and numerical simulations based on physical process models.

The IKBS approach regards the casting field as based on an expert perception of important factors, such as positioning chills and feeders, and determining shape orientation. The similarity metrics we have developed here operate directly on the geometrical features of a shape. Other successful systems, such as Wayland [26] and Clavier [28] operate in similar domains where one would think that being able to assess shape similarity is important, yet these systems still manage to be successful without directly using shape.

Although both Wayland and Clavier use shape, the techniques they use are not directly applicable to our research, mainly because they attempt to solve very different problems. Wayland for instance, is concerned with determining the machine settings of parameter values for die-casting machines, and to do this, the area and weight of a shape are important. In our research, we are concerned with configuring the geometric parameters for casting and this is more heavily dependent on casting geometry. The area and wall thickness, as used in Wayland, doesn't address these deep geometrical features. Similarly, Clavier, one of the early CBR systems is used for curing aerospace parts, performs exact textual matching to past cases, and again is not concerned with geometry.

Numerical modelling software packages are usually based on a finite-element code. The software can predict shrinkage formation in castings to reveal hotspots (such as MAVIS and DIANA) [3], and also to simulate filling and solidification (SIMULAR [4], SOLSTAR [5]). Most of the newer packages allow a casting to be constructed in a CAD system (such as AutoCAD [6]) and swiftly imported, to reveal potential solidification problems. The main practical problem with these systems is that they only give feedback on a proposed design, and do not give advice on the

design itself. The design of castings is a non-trivial problem. An engineer can easily design objects which are extremely similar in appearance, but one can be cast and the other one cannot. The ShapeCBR system currently provides general advice that the engineer can adapt in methoding a new case. Additional information can be gained about a casting by linking in solidification software, such as SOLSTAR, that can provide a visual mapping of the solidification processes to reveal problems such as hot spots and solidification shrinkage.

IKBS systems attempt to support an earlier stage in the design process. Numerous software tools such as those discussed in [7] have clearly demonstrated the usefulness of knowledge-based and other advanced heuristic-based programs for designing castings. Some of the commercial software packages available can calculate the position of feeders (NOVACAST [8]) and also to analyse geometric properties and to give suggestions to improve the design further (AutoCast [9]).

Although many prototype tools have demonstrated the efficacy of CBR in the domain of engineering and design [10-15], there is a scarcity of research for its use in the foundry industry. CBR can play an important role in intelligent casting software. Currently, there is one commercial CBR system [16] called Wayland, which we have briefly mentioned above, for the setting of parameters in pressure die-casting. This research has demonstrated that CBR has an exciting future in casting software.

The main problem for a CBR system is how to retrieve cases, where the retrieval must be based on shape. Although there are other possible search indices, for example the type of casting alloy, weight and general description of part (wheel, sea-gland, valve, engine bearing cap, etc.), these descriptions are too general for accurate retrieval. General classifications of shape components have been proposed; for example, Biederman's *geons* [17]. However, during this research, it became apparent during knowledge elicitation that a decomposition of shapes specific to the casting industry already existed in practice, [7, 18]. The research described here uses a graphical representation of shapes based on this decomposition as a foundation for shape retrieval.

In section 2 of this paper, the graphical representation is explained, and in section 3 similarity measures are proposed. Section 4 gives experimental results for the method based on a trial domain of rotationally symmetric objects. Section 5 discusses the competence of the trial case base, and shows how case base visualisation can assist in refining and maintaining the case base.

## 2 A Graphical Representation

In [7] a decomposition of a shape into a set of joined components was described. The decomposition is a natural one, used over many years by casting design engineers. It is based on a set of component types of significance in casting design. There are 8 main component types including Bar, L, T, X, Taper, Flange, Bespoke-Taper, and Bespoke-T. The componentisation process distinguishes two sets of component type: those that define the structure (L's, T's and X's) and those that join the first set together (bars and tapers). Using this classification, we may abstract a graphical

representation of the structure of any shape S where the nodes are elements of the first set, and the arcs are elements of the second.

As an illustration, consider the rotationally symmetric shape shown in cross section in figure 1. A graph representation of this figure is given in Figure 2.

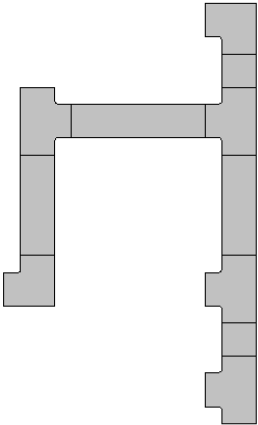


**Figure 1.** A casting, made of 3 component types (Bar, L, T)

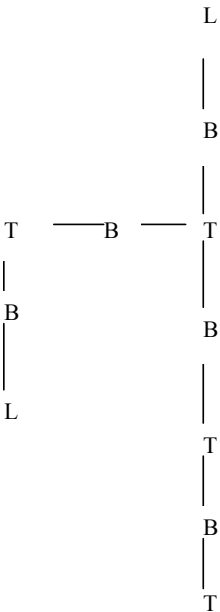


**Figure 2.** Representation of figure 1 as a graph

### 3 Similarity Measures

Retrieval of shapes for casting design is an example of structure based case retrieval, as defined by Gebhardt [15]. For these systems, attributes representing complex structures are difficult to define, and similarity must be derived from structure directly. For the sub class of graphical structures, Gebhardt reviews several retrieval systems. These include clique detection as in the Fabel component Topo [19], largest common subgraph [20] and hamming distance [21].

The research described here at Greenwich, we have used similarity measures based on features extracted from the structural graphs. Perfect similarity between shapes S1 and S2 is obtained when they have identical structural graphs. However for graphs that do not match completely, there are a number of features that can be extracted and compared. Each feature gives rise to a different similarity measure, representing a different case retrieval.

Correspondingly, there are a number of different problems associated with casting a shape, each connected with a different structural feature. Porosity tends to depend on specific local features, whereas machining problems tend to depend on global structure. The approach of this research has been to construct a retrieval tool to investigate the efficacy of the various metrics with respect to different casting design problems. The tool employs a generalised similarity measure  $\sigma(S1, S2)$  between shapes S1 and S2, representing a weighted sum of the similarity measures based on different features extracted from the graphs of S1 and S2:

$$\sigma(S1, S2) = w_{comp}\sigma_{comp} + w_{mcs}\sigma_{mcs} + w_{cycle}\sigma_{cycle} + w_{leaf}\sigma_{leaf} \quad (1)$$

Variation of the weights in this formula allows a general test of retrieval against any given casting problem.

The individual similarity metrics in (1) are defined as follows:

- $\sigma_{comp}(S1, S2)$  is a measure based on the number of component types that are common to the two graphs. If two graphs are nearly identical,  $\sigma_{comp}$  will be close to 1. The length function is defined as  $length(S)$  = number of components in S, and the value of this metric is given by:

$$\sigma_{comp}(S1, S2) = \sum_{Comp} \frac{S'_{comp}{}^2}{length(S1)length(S2)} \quad (2)$$

where  $S'_{comp}$  is the maximal number of common components of a particular type to graphs S1 and S2. Nevertheless, this metric does not take into account of the graph's topology. This is taken care of by the following metric:

- $\sigma_{mcs}(S1, S2)$  is a measure based on the length of the maximum matching subgraph. If two graphs are nearly identical,  $\sigma_{mcs}$  will also be close to 1. This similarity metric is given by:

$$\sigma_{mcs}(S1, S2) = \frac{S'^2}{length(S1)length(S2)} \quad (3)$$

where  $S'$  is the maximal common subgraph of  $S1$  and  $S2$ , i.e. the largest graph which is a subgraph of both  $S1$  and  $S2$ . The problem of finding  $S'$  is related to that of the well-known graph isomorphism problem. For small graphs of up to 10 arcs, a search based on direct comparison of all subgraphs of  $S1$  with those of  $S2$  is possible. For larger graphs a strategy based on a preliminary comparison of node types and degree can help to reduce the search time.

- $\sigma_{\text{ncycle}}$  ( $S1, S2$ ) is based on a count of elementary graph cycles:

$$\sigma_{\text{cycles}}(S1, S2) = 1 - \frac{|\text{ncycles}(S1) - \text{ncycles}(S2)|}{\max(\text{ncycles}(S1), \text{ncycles}(S2))} \quad (4)$$

- $\sigma_{\text{nleaf}}$  is based on a count of leaf nodes, and gives the number of branches to a tree:

$$\sigma_{\text{leaves}}(S1, S2) = 1 - \frac{|\text{nleaves}(S1) - \text{nleaves}(S2)|}{\max(\text{nleaves}(S1), \text{nleaves}(S2))} \quad (5)$$

- $\sigma_{\text{localstati}}$  ( $i = 1, \dots, n$ ) are a set of statistics on local graphical features. For example, an important local feature might be two T's joined together. For this we may take  $\text{localstat}_j$  to represent the number of T's joined by a joining component to another T. We define:

$$\sigma_{\text{localstati}} = |\text{stati}(S1) - \text{stati}(S2)| \quad (6)$$

### 3.1 The *ShapeCBR* System

The *ShapeCBR* software system has been developed at Greenwich to automate the process of matching a given target shape to a case in the Case Base. The case base is populated with cases containing information relevant to real metal casting experience. The information contained in each case relates to both a geometrical description of a real shape and domain specific information about the way that the shape was actually cast. Additionally, some cases may contain general expert advice relevant to casting the shape in a textual form. The system allows the user to retrieve a shape from the case base to match a target case, according to a match on the four contributing features as described in the previous section. Weighing factors can be applied by the user to attach varying importance to each of the similarity measures.

Figure 3 shows an example exercise of matching a target case to a retrieved case from the case base. Advice on positions of feeders and chills is annotated on the picture of the retrieved case.

Early feedback from the use of the system has been promising. The main problems for further development and enhancement of the case base in *ShapeCBR* are seen as:

Is the case base consistent and complete? When is it safe to delete cases? When should we add new cases? To help with these questions we used three tools: the system itself, a multidimensional scatter plot visualisation and a set of benchmark test cases.



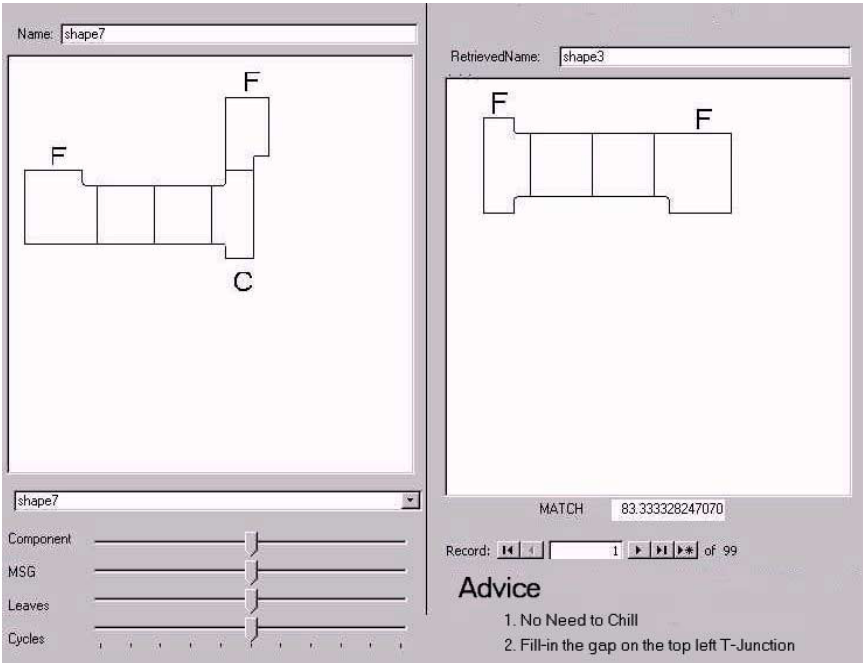


**Figure 3.** Matching a case to a target in the *ShapeCBR* system

The visualisations of the case base, and of the test set, are described in the next section. The distribution of the test set according to the whole case base gives a view of how representative the test cases were, and gives a level of confidence to the weightings used in the system. It may also be used to spot areas where the density of the system is high and where it may be possible to delete some cases.

The system itself can be used to validate whether a particular case is redundant with respect to any test set, by examining the performance measure when the case is removed. However, this is not a safe procedure for a fixed test set – for example, a case base consisting of the 20 test cases themselves would perform 100 % against themselves, but would have little capability for cases outside their scope.

## 4 Measuring Performance

Testing the performance of knowledge based systems has been given much attention in the literature. Guida *et al* [22] argue that performance measures are an essential tool for use in all stages of system development. They are particularly important during system maintenance, where knowledge is added, deleted and modified to effect system adaptation and improvement.

Following the ideas of Guida *et al*, we propose here several performance measures for the CBR system, which give a numerical estimate relative to any benchmark set of problems. In this section we describe the performance measures and give results for a trial case base consisting of 100 cases and tested against 20 new problems. In

section 5 we show how the measures assist in the case base maintenance and refinement.

For the trial conducted here, we have taken the domain of shapes with rotational symmetry: wheels, armatures, cylinders, etc. This domain is coherent from a practical point of view, so that we can attempt to cover it with a limited case base. It is however sufficiently varied to encompass a wide range of casting problems.

Performance of the case base was assessed on several different measures. For a given target the retrieved set should provide the solution to (I) correct orientation, (II) the number and positions of feeders (III) the position of possible chills, (IV) the need for chills. (V) special problems encountered with this shape. For each of these problems, we can score how well the retrieved case presents the answer. For example, consider the target and retrieved shapes shown in Figure 4, which shows retrieved and test cases oriented, as they would be when pouring casting metal into the mould. The character F shows the position of the feeders, and C the chills. A1...A<sub>n</sub> shows presence of advice relating to the need for chills at that position.

In general we can see that there is an obvious visual match between the shapes. Without such a match none of the problems are solved, and we set all the measures to zero. However in this case the match is good enough to indicate the following performance measures:

- (I) 100% - the orientation is correct. The convention in the figure is that the shapes were cast from the top as they are shown. Based on this, either shape will correctly predict the orientation of the other.
- (II) 66% - only one feeder out of two. There is a matching pair of feeders, and one non-matching feeder.
- (III) 100% - both chill positions are correct.
- (IV) 100% - advice as to when chills are needed is correct.

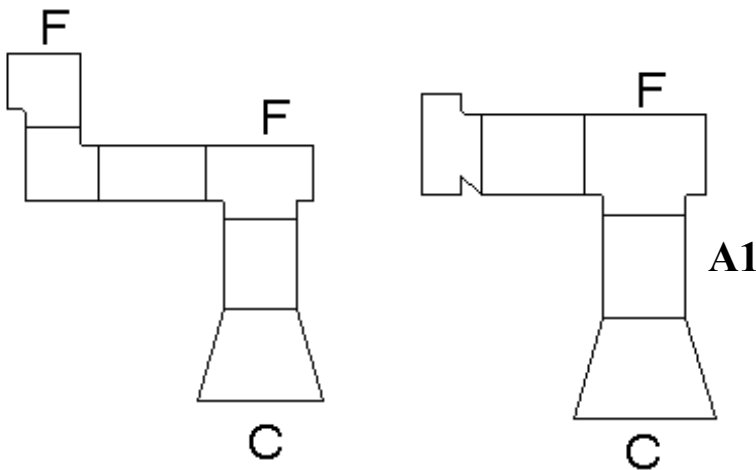


**Figure 4.** Target and Case showing orientation, feeders and chills. Advice A1: “chill this section if length > 4\* breadth”

In situations where no obvious visual match may be made with the nearest case, we can widen the search to retrieve more cases, and leave the user to select the one with the best visual match. In such a mode of operation, the user is allowed to browse the nearest matches to look for the best advice.

4.1 Optimisation of Weights

One use of the performance measure discussed in the first section is in the optimisation of the weights in equation (1). Initially, these were arbitrarily set equal; the idea being that the performance measure can be used to vary the weights to produce optimal performance. A frequently used technique in assessing performance is to measure how well a case base predicts itself. For a case base of 100, this requires us to enter 100 sets of performance data, corresponding to each case as predicted by its nearest match. As weights are changed then often the nearest match will change, and more validation data needs to be added. In reality it proved to be impractical to enter enough validation data to satisfy a wide range of weights required to allow an automatic search on weights for optimum performance.

It was decided to approach the problem incrementally, first selecting a subset of 20 cases, entering validation data for these, and performing an initial search on weights, to provide optimum values:  $w_{Comp0} = 0.6$ ,  $w_{MCS0} = 0.8$ ,  $w_{Cyc0} = 0.4$ ,  $w_{Leaves} = 0.4$ . Next, for weights near to these optimum values validation data was entered for the whole 100 cases, and the optimisation process repeated. Table 1 shows the performance on the whole case base (i) with equal weights, (ii) with optimised weights.

The test set of 20 cases was selected independently and validated by an experienced casting engineer. Figure 5 shows a visualisation of the case base with equal weights, using the method of principal co-ordinates. The distribution of the test cases can also be seen on this plot, and seem to provide a fairly representative spread.

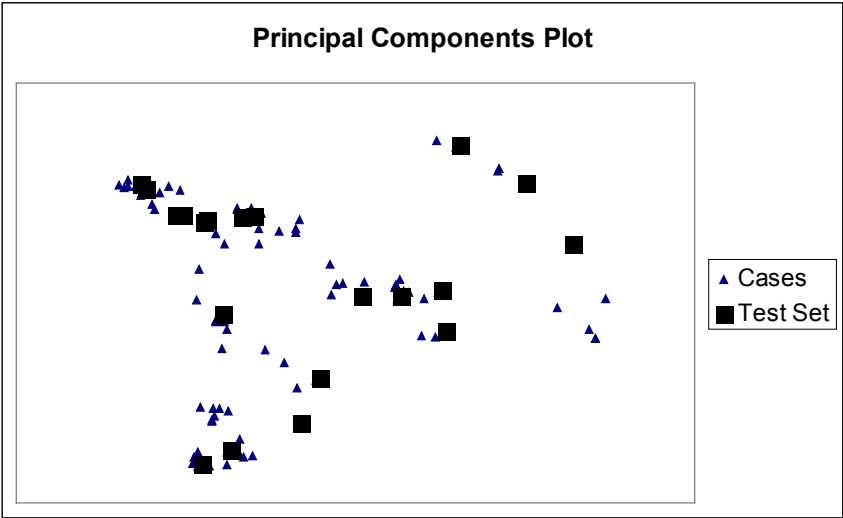


Figure 5. Plot showing distribution of Cases and test set

**Table 1.** Performance for whole case base, as weights are optimised

|                 | Equal weights | Optimised weights |
|-----------------|---------------|-------------------|
| Orientation     | 75.5%         | 79.5%             |
| Feeders         | 74.28%        | 76.08%            |
| Chill positions | 57.28%        | 59.27%            |
| Chill advice    | 82.5%         | 84.5%             |

These performance figures can serve as a benchmark for future maintenance of the case base. We can require that future versions of the case base should perform at least as well on the original 100 cases as the performance figures in Table 1 show. This requirement can help to guide decisions on inclusion or deletion of cases.

## 5 The Competence of the System

An issue of importance in the assessment of CBR systems is that of system competence, that is the number of target problems that a given case base can solve [23, 24, 25]. The performance measures described in section 4 may be used to examine some questions connected with the competence of the case base. These questions are connected with the distribution of cases, i.e. with the problem-solving completeness, density and boundary of the case base. We summarise these questions as follows:

- Completeness: are there any gaps in the case base, i.e. problems that cannot be solved by retrieving near cases?
- Density: is the case base too dense in places? i.e. can we reduce the number of cases without affecting performance?
- Boundary: are there realistic castings representing cases outside our coverage group?
- When should we add new castings to the case base?

There are three tools that we can use to try to answer these questions. Firstly, there are the performance figures discussed above. These can serve as a benchmark for future maintenance of the case base. We can require that future versions of the case base should perform at least as well on the original 100 cases as the performance figures in Table 1 show. This requirement can help to guide decisions on inclusion or deletion of cases.

Secondly there are the principal component visualisations, as shown in figure 5, and three-dimensional scatter plots, which account for more of the distribution of cases. Figure 5 shows a definite boundary to the 100 cases. In fact, we can also see disjoint clusters, such as the shapes with 1 cycle (shown in figure 6). We can also see possible areas where the density of the case base could be high. However, since this is an (important) two-dimensional plot in a 100 dimensional space, care must taken to investigate these regions more carefully. The areas where there are gaps in the plot are difficult to interpret: are there missing realistic cases, or are they outside the boundary? However, the visualisation can give useful information about new cases. If

a new case appears isolated in the middle of a gap, then it is probably a valuable addition to the system.

The third tool at our disposal is the retrieval system itself. As well as providing a tool to investigate further the cases indicated in the plot, it can also be used to find redundant cases. For example, we can run the system as in the performance test, retrieving on each of the cases in turn as targets. If 2 cases are always retrieved as a pair (i.e. one is 2<sup>nd</sup> nearest whenever the other is 1<sup>st</sup> nearest), and they both give the same performance measure on the target, then one of them may be deleted without affecting performance. Using this principle, we can safely eliminate some cases, and reduce density if needed.

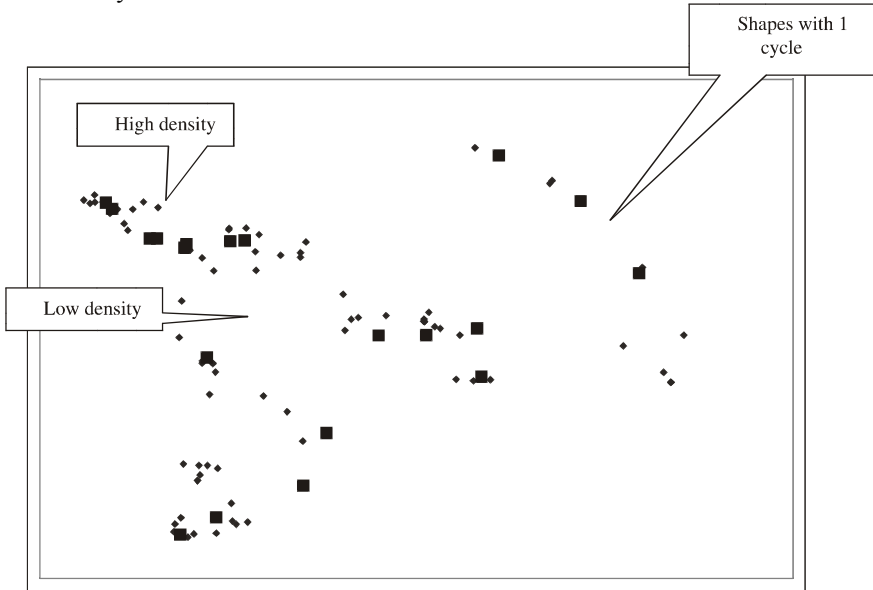


**Figure 6.** Clustering of the case base

## 6 Conclusion

In this paper we have described work on a case based system for the design of metal casting procedures. The key problem addressed by the work is the retrieval of rotationally symmetric shapes. The method proposed is based on a shape componentisation which is particular to the domain of casting problems. The shape componentisation gives rise to a graphical representation of shapes, from which similarity metrics may be abstracted.

The performance of the system has been measured for a sub-domain of rotationally symmetric shapes. A trial system consisting of 100 shapes has been constructed with the assistance of a casting design expert. The performance has been measured with respect to 3 key design decisions, and with respect to the retrieval of associated textual design advice. For the initial system, performance was assessed at between 59% and 84% of expert performance.

The paper also describes work done on the competence of the case base. A representation of the cases as points in an n-dimensional feature space is described, and a visualisation based on the first two principal components is presented. It is shown how refinements suggested by the visualisation may be made to affect case base density and coverage. The performance of the case base after refinement is given and compared to performance before refinement.

The results of these trials are encouraging, and indicate that the method is capable of extension into the full 3D domain of shapes. Future work is planned to extend the trials to wider domains, including general 3D systems. Work is also being planned for the integration of the system with physical modelling systems, such as SOLSTAR, to prototype the casting.

## References

1. Campbell, J. "Castings", Buttnerworth-Heinemann, 1991
2. Jolly, M. "Overview and appraisal of numerical analysis software for the simulation of casting processes", BICTA bulletin 3, 1996
3. Mavis, Diana. Alphacast software Ltd, England
4. Simular. Aluminium Pechiney, Aluval, BP27 38340 Voreppe, France
5. SOLSTAR. Corbett, C. F. "Right first time...with Solstar", Foundry Practice, Number 218, December, 1989
6. AutoCAD. Autodesk Inc. USA.
7. Knight, B; Cowell, D.; Preddy, K. "An object-oriented support tool for the design of casting procedures", In Engineering Applications of Artificial Intelligence, Volume.8, Number 5, pp: 561-567, 1995
8. NovaCast: Sillen, R. "Using artificial intelligence in the foundry", Modern Casting, December, 1991
9. Ravi, B. "Computer-Aided Casting Design - Past, Present and Future", Special Issue of Indian Foundry Journal, January, 1999
10. See p. 374, Marir, F; Watson, I. "Case-based reasoning: a categorized bibliography", The Knowledge Engineering Review, Vol.9: 4, pp: 355-381, 1994
11. Raphael, B; Kumar, B. "Indexing and retrieval of cases in a case-based design system", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 10, pp: 47-63, 1996
12. Hennessy, D; Hinkle, D. "Applying case-based reasoning to autoclave loading", IEEE Expert, pp: 21-26, October 1992
13. Kumar H. S.; Krishnamoorthy, C. S. "Case-based reasoning in bridge design", pp: 197-205, Advances in Computational Structures Technology, Civil-Comp Press, Edinburgh, 1996
14. Yeh, I. "Case-Based approaches for preliminary design of steel building frames", Microcomputers in Civil Engineering, 12, pp: 327-337, 1997
15. Gebhardt, F. "Survey on structure-based case retrieval", The Knowledge Engineering Review, Vol.12: 1, pp: 41-58, 1997

16. Price, C. J.; Peglar, I. S.; Ratcliffe, M. B.; McManus, A. "From troubleshooting to process design: closing the manufacturing loop", *Lecture Notes in Artificial Intelligence*. 1266, Case-based reasoning research and development. Springer-Verlag, 1997
17. Biederman, I.; Hummel, J. E.; Cooper, E. E.; Gerhardstein, P. C. "From image edges to geons to viewpoint invariant object models: A neural net implementation", Applications of Artificial Intelligence X: Machine vision and robotics, 1992
18. Wlodawer, R "Directional solidification of steel castings", Pergamon, Oxford, 1967
19. Coulon, C. H. "General geometric and topological retrieval and adaptation (Topo)". In Borno, K, editor, *Modulus for design support*, Vol. 35 of Fabel report, pp: 33-445, GMD, Sankt Augustin, 1995
20. Tammer, E. C; Steinhofel, K; Schoner, S; Matuschek, D. "Anwendung des Konzepts der strukturellen Ähnlichkeit zum Fallvergleich mittlerer Term- und Graph-representationen", Fabel report, 38, GMD, Sankt Augustin, 1995
21. Bunke, H; Messmer, B. T. "Similarity measures for structured representations". In Burkhard, H. D. and Lenz, M, editors, *Topics in Case-Based reasoning: First European workshop, EWCBR-93, selected papers; Lecture notes in Artificial intelligence*, 837, pp: 106-118, Springer-Verlag, 1994
22. Guida, G; Mauri, G. "Evaluating performance and quality of knowledge-based systems: foundation and methodology", IEEE Transactions on Knowledge and Data Engineering, 5, 2, pp: 204-224, April 1993
23. Preece, A. D. "Towards a methodology for evaluating expert system", Expert Systems, 7(5), pp: 14-20, 1992
24. Racine, K; Yang, Q. "Maintaining unstructured case." Proceedings of the 2<sup>nd</sup> International Conference of Case-Based Reasoning, pp: 553-564, R1, USA, 1997
25. Smyth, B. "Case-Based Maintenance", *Proceedings of the 11<sup>th</sup> International Conference on Industrial & Engineering Applications of AI & Expert Systems*. Springer-Verlag, 1998
26. Price, C. J.; Peglar, I. S. "Deciding parameter values with case-based reasoning", *Progress in case-based reasoning* (ed. Ian D. Watson), Springer-Verlag, pp: 121-133, 1995
27. See page 123. Price, C. J.; Peglar, I. S.; Bell, F. Case based reasoning in the melting pot, Int. Journal of Applied Expert Systems, vol.1, no.2, pp: 120-133, 1993
28. Hennessy, D; Hinkle, D. "Applying case-based reasoning to autoclave loading", IEEE Expert, pp: 21- 26, October 1992

# Personalised Route Planning: A Case-Based Approach

Lorraine McGinty and Barry Smyth

Smart Media Institute, Department of Computer Science,  
University College Dublin, Belfield, Dublin 4, Ireland  
{Lorraine.McGinty,Barry.Smyth}@ucd.ie

**Abstract.** Automatically generating high-quality routes using real map data is difficult for a number of reasons. Real maps rarely contain the sort of information that is useful for constructing high quality routes. In addition, the notion of “route quality” is difficult to define and is likely to change from person to person. In this sense the automatic construction of high-quality routes that match the preferences of individuals is an example of a weak-theory problem, and therefore well suited to a case-based approach. In this paper we describe and evaluate a case-based route planning system that is capable of efficiently generating routes that reflect the implicit preferences of individual users.

## 1 Introduction

Route planning has always provided a rich vein of research for the artificial intelligence community, however, to date the majority of this research has focused on the generation of optimal routes using the so-called shortest path algorithms to minimise distance travelled. A variety of heuristic variations have also been introduced to reduce the cost of these algorithms. In this sense route planning is an example of a strong-theory problem (since complete information about the geometric properties of a map are known) and therefore is amenable to a traditional first-principles planning approach.

However, recently researchers have begun to look at a related problem that does not benefit from a complete domain model. In general, the problem of generating high-quality and realistic routes means more than planning the shortest-path from a start location to an end location [2,3,5,6,7,8]. It is a weak-theory problem because of a number of complicating factors that introduce a large degree of uncertainty into the planning process. First and foremost, digital maps rarely encode the sort of information needed to generate high quality route plans, information such as traffic patterns, road quality, or congestion information. Secondly, the concept of a good route depends very much on the user in question; in other words, different users will prefer different routes according to their general preferences, driving experience and knowledge of the mapped area. Thirdly, the factors that influence an individual’s preferences may not even be known.

The starting point for our work is the assumption that it is not possible to factorise the internal route-planning preferences of a given user; in other words,



we cannot characterise the preferences of a user by using a well-defined set of features. Our goal is to develop a system capable of responding to the implicit preferences of a user by drawing on their prior route planning experiences.

In this paper we describe Turas<sup>1</sup>, a case-based route planning system that is personalised for the needs of individual users [4,9,10,11]. Each user's route planning preferences are represented as a collection of previous route cases, and reusing relevant sections of multiple cases generates new routes. We describe and evaluate our route planning algorithm and demonstrate that it is significantly more efficient than traditional algorithms, such as A\*, while generating routes of superior quality. We also introduce a new fast-indexing technique that reduces the retrieval cost for large case-bases, without compromising solution quality.

## 2 Related Work

Related work on the problem of route planning covers a number of research areas. Traditional approaches have developed from graph theory and use shortest-path algorithms such as Dijkstra's method or A\*. However, these algorithms are computationally expensive and wasteful in their search strategies, often considering route sections that could not possibly form part of a realistic solution.

Recently a number of researchers have proposed knowledge-based and experienced based route-planning algorithms. Smyth and Cunningham [9,10] describe a simple case-based route planner while Branting and Aha [1] describe a more sophisticated hierarchical case-based route planner that generates new routes at multiple levels of abstraction. While these attempts have focused on artificial route planning problems they have served to demonstrate the computational benefits associated with these alternative route-planning strategies.

Haigh and Veloso [2,3] describe a case-based route planner that, like Turas, constructs new target routes by reusing sections from previous similar route cases. The focus of their work is on the development of sophisticated case indexing, retrieval, and adaptation algorithms. They show that a case-based approach to route planning can result in significant speed-up over brute-force search techniques, such as A\*, without significant loss of solution quality.

Liu [5,6] describes a route planning system that combines geographical knowledge with case-based reasoning and brute-force search. Liu describes how geographical knowledge can be used to isolate the search for useful route segments to a local map region. The approach makes the strong assumption that users prefer routes that follow major roads, and the planning algorithm explicitly seeks out major roads to form the backbone of a target route. We agree that this assumption often holds, but we do not agree that it is wise to hard-code it into a route-planning algorithm. Certain users may wish to avoid the major road networks, for example. The point is, that different users can have very different preference models, often grossly at odds with any expected norm, and these users will not be well-served by such a route planner.

---

<sup>1</sup> Turas is a gaelic word meaning journey or trip.

Rogers and Langley [8] describe a route planning system that, like Turas, acknowledges the fact that the preference models used by real users can deviate significantly from simple distance-based models. They describe a system that can learn user preferences by processing feedback provided by users over a number of sessions. During each session users are asked to express a preference between recommended routes and this feedback is used as the training data for a perceptron-style training algorithm. An important difference between this work and our own is that the authors assume a fixed user preference model - they propose relevant features such as route length, driving time, and turn angles. Each user's preferences are expressed as a set of weights for these features. Our position is that it may not be possible to recognise that such features are important to a given user, and while the proposed features seem reasonable we argue that many more features are undoubtedly relevant but unknown. Moreover, we suggest that different sets of features may be relevant to different users. Our approach does not assume a fixed preference model, which, we argue, makes it more generally applicable for a wider range of route-planning scenarios.

We conclude that modern route finding applications demand a more sophisticated algorithm than traditional approaches. New approaches introduce knowledge-based heuristics and experience in order to guide the planning process and there is a growing understanding that the generation of high quality routes must take account of the explicit and implicit preferences of individuals. We argue that not only must route planners generate routes that are personalised with respect to individuals; they must do so without making strong assumptions about the type of preference model that a given user might use.

### 3 Personalised Route Planning in TURAS

Turas is a Java-based Web service and is designed for use as a general purpose personalised route planner; at the moment the system is designed to plan routes around Dublin city, but of course it can be easily adapted for other regions.

The system allows users to specify new route planning problems by selecting start and end locations from a digital map display and presents a graphical and textual representation of the resulting route to the user. Importantly, the system is personalised for the needs of individual users, generating routes that reflect the learned preferences of a given user, rather than more general routes that aim to minimise some global cost function. As such it requires users to register with the site so that user profiles (encoding their implicit route preferences) may be maintained and used during the route planning process.

In the remainder of this section we will outline the overall system architecture and describe the case-based route-planning algorithm used to generate personalised route plans that are appropriate for individual users.

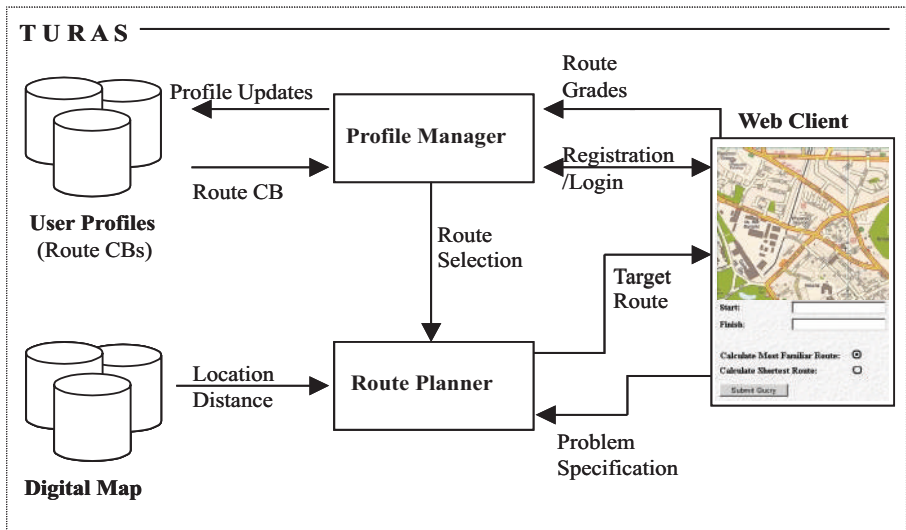


Fig. 1. The Turas system architecture

3.1 System Architecture

The Turas architecture consists of the following components:

**Digital Map:** The map database contains standard electronic map information including basic location, street name, and distance information. We do not assume that any additional information, such as traffic patterns or road quality information is available.

**User Profiles:** Each user profile is composed of a set of previously graded routes - each profile element contains a route plan (as a list of connected road segments from a specified start location to an end location) and an assigned grade. Essentially each profile acts as a local case-base of route planning experience for a given user, and is used by the case-based route-planning algorithm to generate personalised route plans.

**Route Planner:** The route planner uses the case-based planning algorithm described in Section 3.3 to generate a plan that solves the specified route problem. In Section 3.3 we will argue that these plans are personalised for the particular target user, because they are based on the route planning experiences of that user, encoding his or her implicit travel preferences.

**Profile Manager:** The profile manager is responsible for user registration and logging and, importantly, for managing the grading information that users provide as a result of ongoing route planning sessions. In particular, when the system generates a new route for a user, the user is given the opportunity to grade or rate this route as positive or negative, and this information is used to update the user's profile.

### 3.2 Representational Issues

The map used in our work is a complete map of Dublin city (900 junctions and 2500 road segments where each individual street is typically made up of a number of road segments). The map database contains a complete description of the map including road names, junction co-ordinates, and connectivity information, and is typically of the sort of information that is normally available in a standard digital map. Crucially, the map lacks the sort of information that is needed for high-quality route planning including, traffic flow information, road quality information, neighbourhood designations (residential, commercial), etc. It is this lack of information that makes high-quality route planning so difficult in reality.

Each route is made up of a sequence of road segments joining a start location (junction) to an end location (junction). The simplest representation for the route cases is a simple list of the road segments in a route - the road segments themselves act as the case indices during retrieval. Thus, each case corresponds to a single route and contains no other information except for the sequence of road segments linking the start and end locations.

In the next section we will assume this simple case representation when describing our case-based route-planning algorithm. However, the representation does suffer from severe efficiency shortcomings and in Section 3.4 we will describe a more abstract representation and fast-indexing algorithm that can significantly improve retrieval cost without compromising retrieval quality.

### 3.3 A Case-Based Route Planning Algorithm

The case-based route-planning algorithm is outlined in Fig. 2. Each new route is generated recursively by retrieving and adapting multiple cases to fit the current problem at hand. Each recursive call of the algorithm attempts to solve part of the route problem by reusing a case segment such that the remaining uncovered distance is maximally reduced. If a suitable case cannot be found, or if the distance between the current start and end locations is below a set threshold, then the standard distance-based A\* planning algorithm is used to complete the route.

An example, based on a Dublin route-planning problem from X to Y, is shown in Fig. 3(a and b). The first recursive call selects a case C1 such that the remaining uncovered distance  $|X, X1| + |Y1, Y|$  is minimal over all cases; X1 and Y1 are those junctions in C1 that are closest to X and Y, respectively. C1 is adapted by deleting those route segments from the start junction of C1 to X1 and from Y1 to the end junction of C1; these dropped segments are shown as dotted lines in Fig. 3(a). This produces two new route planning sub-problems, one from X to X1 and one from Y1 to Y, and subsequent iterations of the algorithm address each of the sub-problems in a similar fashion by selecting two further cases, C2 and C3, as shown in Fig. 3(a). Fig. 3(b) illustrates the final route, which is made up of three separate case sections plus a number of connecting sections produced by A\* (dotted lines in Fig.3(b)).

```

RoutePlan(start, end, CB, threshold)
1  If Dist(start,end) < threshold then
2    route  $\leftarrow$  A*(start,end)
3  Else if
4    case  $\leftarrow$  RetrieveCase(start, end, CB)
5    If case then
6      section  $\leftarrow$  AdaptCase(start, end, case)
7      route  $\leftarrow$  RoutePlan(start, Start(section),CB,threshold)
          + section + RoutePlan(End(section),end,CB,threshold)
8    Else route  $\leftarrow$  A*(start,end)
9  End if
10 Return(route)

RetrieveCase(start, end, CB)
10 For each case C $\in$ CB
11   C.X'  $\leftarrow$  junction in C with min Dist(start,X')
12   C.Y'  $\leftarrow$  junction in C with min Dist(Y',end)
13 End For
14 C  $\leftarrow$  case with min Dist(start,C.X') + Dist(C.Y',end)
15 Return(C)

AdaptCase(start, end, C)
16 C.X'  $\leftarrow$  junction in C with min Dist(start,X')
17 C.Y'  $\leftarrow$  junction in C with min Dist(Y',end)
18 section  $\leftarrow$  road segments in C from C.X' to C.Y'
19 Return(section)

```

**Fig. 2.** The case-based route planning algorithm

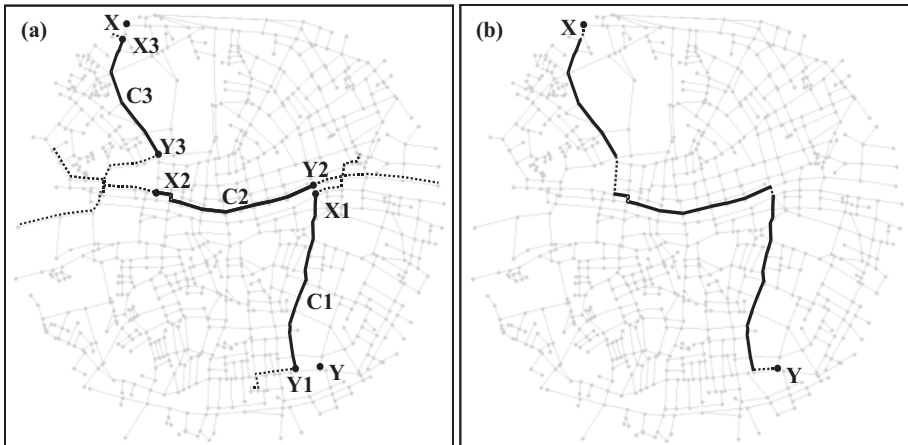


**Fig. 3.** An example case-based route planning session showing the generation of a target route from three separate case sections (a) and the resulting final route (b)

### 3.4 A Fast Indexing Technique for Efficient Case Retrieval

The simple case representation scheme introduced in Section 3.2 is problematic in the sense that it leads to an expensive case retrieval process with a complexity of  $O(mn)$ , where  $m$  is the number of cases in the case-base and  $n$  is the average number of junctions per case. Essentially every junction of every case must be compared to the target start and end locations in order to compute the distance values needed by lines 11 and 12 of the planning algorithm. In this section we describe a more efficient case indexing and retrieval technique.

Instead of representing cases as sequences of route junctions, and using these junctions as the indices during retrieval, we propose a simple abstraction method whereby each case route is viewed as a straight-line connecting the case start and end locations - we call this the case line segment; see also [2,3] for related ideas. For example, Fig. 4 shows a route case ( $C$ ) from start location  $S_C$  to end location  $E_C$  and its corresponding case line segment ( $C'$ ).

During similarity assessment we need to compute two distance values: the distance from the target start ( $S_T$ ) and end ( $E_T$ ) locations to their nearest case junctions. We propose estimating these values by computing the distance from the target start and end locations to the case line segment. We do this by computing the length of two line segments that connect the target start and end locations to the case line segment. These line segments are called the start intersection line (SI) and the end intersection line (EI), respectively.

To construct each intersection line we construct a line from the relevant target location that is perpendicular to the extended case line (that is, the case line extended to infinity at either end). By solving the two line equations (the extended case line and the perpendicular line from the target location) we can compute where the perpendicular line intersects with the extended case line. We call this point the crossing point and it is our estimate of the nearest case junction to the relevant target location. If the crossing point lies on the case line segment, then the distance of the target location from the case line is the distance of the target location from this crossing point; that is, the perpendicular distance from the target location to the case line. If on the other hand, the crossing point does not lie on the case line segment, then the distance from the target location to the case line is the distance from the target location to the appropriate case line terminal point; if we are looking at the target start location then the appropriate case line terminal point is the case line start location.

An example is shown in Fig. 4 for a case  $C$ , from  $S_C$  to  $E_C$  and with a case line  $C'$ , and for target start and end locations  $S_T$  and  $E_T$ . The perpendicular line from  $S_T$  crosses  $C'$  (at  $x$ ) while the perpendicular from  $E_T$  does not; it crosses the extended case line at  $y$ . Therefore, the distance from  $S_T$  to  $C'$  is the distance from  $S_T$  to  $x$ , while the distance from  $E_T$  to  $C'$  is the distance from  $E_T$  to  $E_C$ , the end point of the case line segment. The similarity of the target problem to  $C$  is the sum of these two distance values.

Thus, the similarity of a case and a target can be efficiently calculated as the sum of the lengths of the intersection lines for  $C$  (that is,  $|SI| + |EI|$ ), without the need to examine each case junction in turn. Retrieval then proceeds to locate

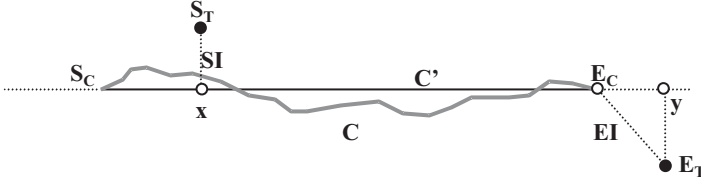


**Fig. 4.** The similarity between a case,  $C$ , and a target problem represented by a start ( $S_T$ ) and end ( $E_T$ ) location, is the sum of the lengths of  $SI$  and  $EI$

the case that minimises this distance sum; retrieval complexity is thus reduced to  $O(m)$  in the size of the case-base. Of course, this similarity metric does not guarantee optimal results because the intersection lines may not intersect with the case line at real junctions. And if case routes are not well represented by their corresponding case lines, then the retrieval method may select a case that is not the closest to the target start and end locations. However, we suggest that the case line approximation will hold well for most route planning situations and that the efficiency benefits it offers far outweigh any reduced retrieval accuracy.

### 3.5 Discussion

As mentioned previously, traditional approaches to route planning are based on strong assumptions about the sort of factors that are important in planning high-quality routes. For example, many approaches assume that distance travelled is all important [1], while others propose cost functions or preference models that are based on pre-determined features such as time of day, or the road quality, or the scenic value of a route, or even the number of turns in a route [5,6,7,8].

Our approach, as described above, makes no strong assumptions about the sort of preference models that people might use when planning or evaluating routes. In fact, we assume that such knowledge is not explicitly known or available. We do however assume that such information is implicitly encoded within any routes that have been liked by a user in the past [3]. In other words, we suggest that any routes that have been liked by a user in the past must have been preferred because they conform to the preference model of that user. Moreover, we suggest that it is possible to combine high quality route sections without loss of quality and thus that the above algorithm will construct new routes that also conform to a given user's preference model by combining and reusing high quality routes sections from the target user's case-base.

## 4 Experimental Evaluation

In this section we evaluate our case-based route planning approach focusing on the efficiency characteristics of the algorithm and the quality of the resulting routes.

## 4.1 Set-up

As part of our evaluation we compare two versions of our case-based planning algorithm, the standard version without the fast indexing (CBR-Standard) and the version with fast indexing (CBR-FI), to the traditional A\* (distance-based) algorithm. However, we first need to produce a case-base that reflects the cost function of an individual user, a cost function that is not purely distance based.

Unfortunately, real users, as test subjects, are hard to come by and were not available for the study at hand. Instead we chose to simulate users by defining artificial cost functions and generated case-bases that conform to these functions; this is similar to the “dummy profile” strategy described in [7,8]. We define a user cost function by assigning random weights to the road segments of our digital map, and the cost of an individual road segment is computed according to the cost function shown in (1).

$$Cost(segment) = length(segment) * weight(segment) \quad (1)$$

One can view these weights as being inversely proportional to the ‘desirability’ of the road segment for a given user, where desirability is based on some complex and hidden user preference model. Road segments with a high weighting have a low desirability and present with a higher cost than similar length road segments with a low weight (high desirability).

We can use this type user-specific cost function (where each user has a different set of road-segment weights) to generate arbitrarily large user profile case-bases by generating solutions to random problems using A\* with the user’s actual cost function. This will guarantee to generate a case-base of routes that minimise the user’s cost function - the sum of the segment costs of a given route as in (1).

## 4.2 Method

We generate profile case-bases of varying sizes (50 - 350 cases) for a hypothetical user according to the method described above. We further generate a set of 60 unseen random target problems. Each target problem is solved using our three route-planning algorithms: the standard distance-based A\* approach; the standard CBR approach (without fast indexing); and the CBR approach with fast indexing. The case-based approaches generate routes by using each of the different sized case-bases, whereas the A\* approach generates routes in the normal way by using a standard distance-based Euclidean evaluation function. It is vital to note that none of these methods have access to the user’s implicit cost function as defined in (1), but we argue that the CBR approaches will nonetheless generate routes that tend to respect the user’s cost function by reusing high quality route sections from preferred cases. As a benchmark, against which to judge our newly produced routes, we also generate the optimal route for each target problem by using A\* along with the actual cost function shown in (1) - this optimal route represents the ideal route for the given user.

To evaluate the efficiency of the various approaches we measure the mean problem solving time overall target problems for each of the different algorithms;



this includes measuring the mean problem solving time of the CBR systems for each of the different case-base sizes.

We evaluate the quality of the resulting routes in two ways. First we compare each target solution to the optimal benchmark route and count the mean number of shared (overlapping) road segments. Thus, an overlap of 70% means that the route in question contains 70% of the road segments used by the optimal route. In addition, we also evaluate the familiarity of the target routes by computing the percentage of road segments in a route that are found in cases in the case-base. Thus, a familiarity value of 80% means that 80% of the road segments in the route are known to the user in question (in the sense that they are in cases in her user profile and thus have been travelled before); a number of researchers have pointed out in the past that route familiarity must be a vital component of any route quality measure (e.g., [2,3]).

### 4.3 Results

The results are shown in Fig. 5(a-c) as graphs of mean planning time, mean route overlap, and mean familiarity versus case-base size; for the planning time and overlap graphs the distance-based A\* results present as a straight line since these measures are independent of case-base size.

**Efficiency** - Fig. 5(a) demonstrates the efficiency characteristics of the distance-based A\*, CBR-Standard, and CBR-FI algorithms. Clearly the results are very positive for the case-base variations. The standard case-based approach without fast indexing achieves a speed-up of between 2 and 8 when compared to the distance-based A\* technique, however planning time is seen to increase significantly with case-base size. In contrast, the case-based approach with fast indexing achieves a speed-up of between 10 and 36 compared to distance-based A\*, and is seen to scale very well with increasing case-base size. Projecting the case-based graphs for larger case-bases we find that, while the planning time of the standard case-based approach will match distance-based A\* for case-bases of approximately 650 cases, the CBR-FI method will offer reduced planning times (compared to distance-based A\*) for case-bases of up to 5500 cases. In summary then, the CBR approaches achieve superior efficiency results when compared to distance-based A\* - the question that remains is whether these efficiency gains are at the expense of route quality?

**Optimal Overlap** - Fig. 5(b) presents the results for the first of our quality measures, namely the mean overlap between target routes and the optimal routes. The distance-based A\* routes have an overlap value of 39.65% meaning that these routes are missing more than 60% of the road segments used in the optimal route. In contrast, and as predicted, the case-base approaches achieve superior overlap values, ranging from 50% for small case-bases to 70% at 350 cases. Clearly, the case-based approaches benefit from the availability of larger case-bases (overlap tends to increase with case-base size), which is to be expected as larger case-bases will tend to provide the planner with better access to the user's implicit preference model, by contributing more relevant cases to the planning process. Interestingly, there is very little difference between the overlap

characteristics of the standard CBR approach and the fast indexing approach. This indicates that the impressive efficiency characteristics of the fast indexing approach do not have an obvious adverse effect on solution quality.

**Familiarity** - Fig. 5(c) presents the results for our second quality measure as a graph of mean route familiarity versus case-base size. Notice that this time the distance-based A\* result is affected by case-base size since as the case-base grows it is natural that more road segments will be covered by cases and thus the base-line familiarity will increase; we also show the familiarity results for the optimal routes. Again the results are very positive with the case-based methods producing routes that are consistently more familiar to the user than the routes produced by distance-based A\*. Interestingly, we notice that the case-based routes are more familiar to the user than the optimal routes at small case-base sizes, thus compensating for the lower overlap values found for these routes in the previous experiment.

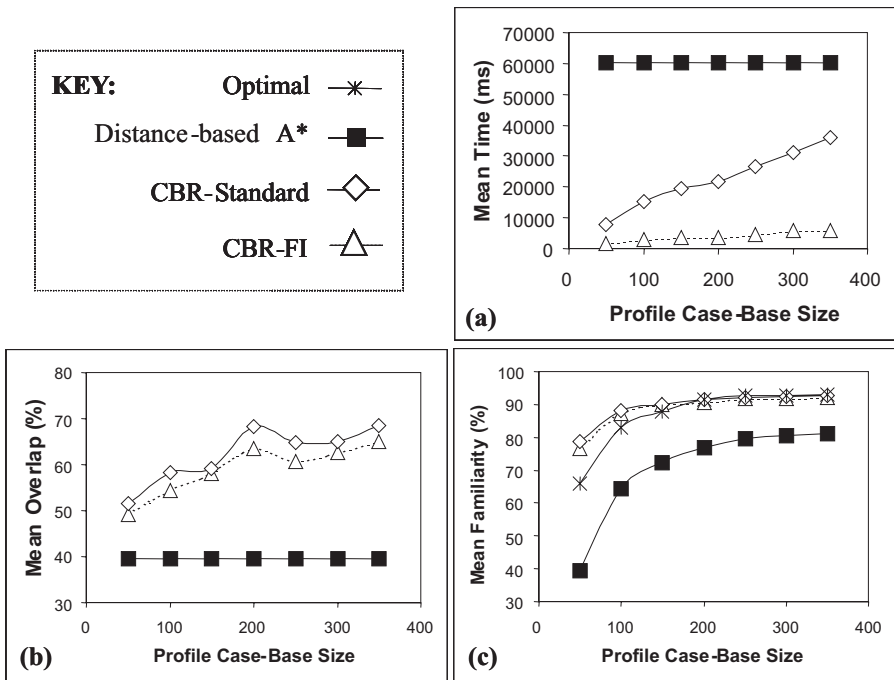


**Fig. 5.** Experimental results for the efficiency and quality characteristics of the case-based route planning methods as compared to a traditional A\* approach

## 5 Conclusions

In summary, Turas, is a Web-based, personalised route planner that is capable of learning about the route planning preferences of individual users and of generating routes that reflect these preferences without imposing assumptions about the nature of these preferences. We have described a new case-based route planning technique that makes no strong assumptions about the preference models employed by users, and yet generates high quality user-specific routes. A fast case-indexing technique for use with this planner has also been developed.

Our initial evaluation results show that the case-based planner is capable of generating high quality routes, which reflect the implicit preferences of a user, but in a fraction of the time taken by traditional methods such as A\*; in fact we have seen that the routes produced by the case-based methods are of higher quality than those produced by A\*. We have also shown that the case-based planner combined with the fast indexing technique provides a scalable retrieval solution for case-based route planners. Our future work will concentrate on the large-scale evaluation of Turas using both artificial and real users.

## References

1. Branting, L. and Aha, D.: *Stratified Case-Based Reasoning: Reusing hierarchical problem solving episodes*, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1995, pp. 384–390. 432, 438
2. Haigh, K., Shewchuk, J., and Veloso, M.: *Exploiting Domain Geometry in Analogical Route Planning*, Journal of Experimental and Theoretical Artificial Intelligence 9 (1997), 509–541. 431, 432, 437, 440
3. Haigh, K. and Veloso, M.: *Route Planning by Analogy*, Proceedings of the International Conference of Case-Based Reasoning, Springer-Verlag, 1995, pp. 169–180. 431, 432, 437, 438, 440
4. Kolodner, J. (ed.): *Case-Based Reasoning*, Morgan Kaufmann, 1993. 432
5. Liu, B.: *Using Knowledge To Isolate Search in Route Finding*, Proceedings of Fourteenth International Joint Conference on Artificial Intelligence, 1995, pp. 119–124. 431, 432, 438
6. Liu, B.: *Intelligent Route Finding: Combining Knowledge, Cases and An Efficient Search Algorithm*, Proceedings of the 12th European Conference on Artificial Intelligence, 1996, pp. 380–384. 431, 432, 438
7. Rogers, S., and Fiechter, C.: *A Route Advice Agent that Models Driver Preferences*, Proceedings of the American Association of Artificial Intelligence Spring Symposium on Agents with Adjustable Autonomy, 1999, pp. 106–113. 431, 438, 439
8. Rogers, S., and Langley, P.: *Personalized Driving Route Recommendations*, Proceedings of the American Association of Artificial Intelligence Workshop on Recommender Systems, 1998, pp. 96–100. 431, 433, 438, 439
9. Smyth, B. and Cunningham, P.: *The Utility Problem Analysed: A Case-Based Reasoning Perspective*, Proceedings of the American Association of Artificial Intelligence Spring Symposium on Agents with Adjustable Autonomy (I. Smith and B. Faltings, eds.), Springer-Verlag, 1996, pp. 392–399. 432

10. Smyth, B. and Keane, M.: *Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning*, Artificial Intelligence **102** (1998), 249–293. 432
11. Watson, I. (ed.): *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann, 1997. 432

# A Case-Based Approach to Image Recognition

Alessandro Micarelli<sup>1</sup>, Alessandro Neri<sup>2</sup>, and Giuseppe Sansonetti<sup>1</sup>

<sup>1</sup> Dipartimento di Informatica e Automazione, Università degli Studi “Roma Tre”  
Via della Vasca Navale, 79, I-00146 Roma, Italia

{micarell,gsansone}@dia.uniroma3.it

<sup>2</sup> Dipartimento di Ingegneria Elettronica, Università degli Studi “Roma Tre”  
Via della Vasca Navale, 84, I-00146 Roma, Italia

neri@ele.uniroma3.it

**Abstract.** In this paper we present a case-based approach to the recognition of digital images. The architecture we propose is based on the “wavelet transform” that has been used for the representation, in the form of old cases, of images already known to the system. The paper also presents our report on a case study in the field of “mobile robots”. The described system is capable of analyzing maps obtained from the sensors of a robot, and classifying them as one of the possible “objects” present in the environment in which the robot navigates. The first results we have obtained are encouraging and support the choice of the case-based approach to image recognition using the wavelet transform as a tool for image representation and analysis.

## 1 Introduction

The purpose of this study is to show how the case-based philosophy may be successfully applied to fields such as the one of image recognition, where, with the exception of rare but significant instances [5,6], the limited availability of literature concerning this subject-matter could misleadingly suggest that CBR is not suitable for possible application in this field. Through our work, we wish to emphasize the advantages obtained in tackling image recognition problems by exploiting a number of already resolved cases, appropriately represented.

In principle, the ideal situation in respect to an image representation would be to emphasize – inside the image-signal – those very zones that are the focus of human vision interest. In this respect, extensive studies have demonstrated that the first elements that attract our attention at a perceptive level are lines and edges, i.e., elements that can be defined by way of measurable quantities on the basis of values given to image signals. Our choice in respect to the representation of the images to be subsequently analyzed, is based on the theory of wavelet transform. During the past decade, the wavelet transform theory has obtained considerable success in various application fields. In fact, numerous signal processing applications relate to this theory, such as image multi-resolution encoding, sub-frequency band encoding for video signal compression and the analysis of non-stationary signals such as geophysical and bio-medical ones.

Our paper is organized as follows: in Section 2, we illustrate a generic type of case-based architecture that we propose for the recognition of a vast range of images. In particular, we describe the representation technique for images in the library of cases and the retrieval algorithm operating on such library in respect to the analysis of new images. In Section 3, we describe a case study relating to the application of our recognition approach to objects of interest in the field of mobile robots. Our final remarks are given in the concluding section.

## 2 The Wavelet-Based CBR Architecture

The problem we intend to address concerns the recognition of an image represented in digital format and its classification under one of the categories of a set of predetermined categories, each of which is linked to an “object” of interest. As we already mentioned earlier, the idea of the approach presented herein consists in taking advantage of the tools officered by the wavelet transform theory, appropriately adapted for use in a case-based architecture. Figure 1 illustrates a diagram that summarizes the mechanism we propose. Let’s assume we have a new image to classify, in bitmap format (termed “new digital image” in the figure), the initial operation performed relates to the extraction of “features” relevant for the purposes of the image recognition. The result of the pre-processing will be the representation of the image based on the wavelet transform. This representation constitutes the “new case” of the proposed CBR system. The retrieval module shown in the figure will effect a research in the case library containing the old cases, with a *<problem representation, solution>* structure, which in this specific case will be *<image representation, object represented by the image>*. The solution given in the old case can therefore be seen as a pointer to a new library, which we can name “Library of Objects”, containing the objects (i.e., “categories”) that could be present in the images to be analyzed. The “Recognized Object” is at this point taken into consideration by the environment where the system is operating.<sup>1</sup> This object, which constitutes the old solution of the case retrieved from the Case Library, will also be considered as the solution of the new problem (basically, there is no need for an adaptation of the old solution to suit the new case) and if the environment accepts it, the pair *<New Image, Recognized Object>* can be inserted as a new case in the Case Library. The most problematic aspects to be worked out are therefore the actual modalities for the representation of images as cases and the definition of a similarity metric to be used for the retrieval phase in respect to the Case Library.

In principle, given a set of known cases, a simple approach to retrieve the one that better matches the situation under examination could be based on the computation of the cross-correlation between the new image and templates

---

<sup>1</sup> For a better understanding of this idea, the environment of the case-study presented in Section 3, concerning a vision problem in respect to mobile robots, can be seen as constituted by the robot and the human expert who is supervising the training of the robot itself.

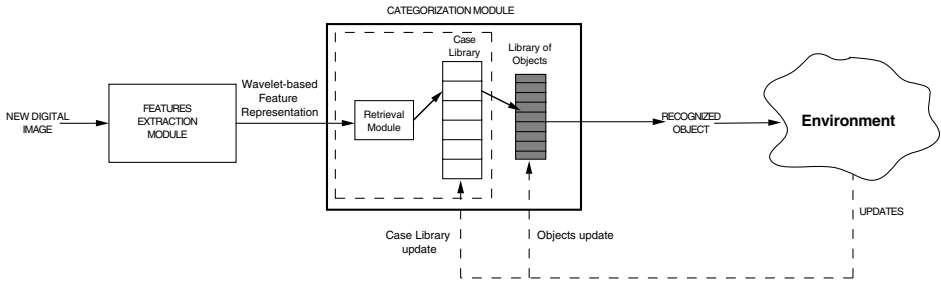


**Fig. 1.** CBR Approach to Image Recognition

extracted from the old cases. However, although conceptually simple, this approach is rather ineffective. In fact, the ability of handling cases derived from those contained in the library by applying to them simple geometric transformations, such as translation, rotation or scaling, would require the computation of the said metric for infinitely many variants. In addition, direct evaluation of the cross-correlation function would imply storage and on-line processing of the related images. To manage this computational burden, a viable approach for case indexing could rely on the use of some kinds of “invariant” signatures, investigated in the past decades in the context of classical pattern recognition [8,12]. The major class of rotation invariant image classification techniques, originally introduced in the field of optical processing (see for instance [8,23]), is based on the extraction of the dominant circular harmonic components obtained expanding a given image in its angular Fourier’s series [12,15,18,9,10,11,24]. This circular harmonic decomposition can be further combined with scale invariant representations like the Fourier-Mellin Transform (FMT) to devise rotation and scale invariant pattern recognition algorithms [22]. To preserve the ability of the system of inferring the current situation based on the whole set of collected cases, at the expenses of a reasonable computational complexity, in our architecture we resort to a general purpose rotation and scale invariant indexing tool based on the cited mathematical representations, proposed in [11], and [12]. In essence, the index associated to each pattern stored in the case dictionary is constituted by the dominant components of the Laguerre-Gauss Circular Harmonic Wavelet (CHW) decomposition, computed by means of a bank of linear operators.

Since retrieval requires the evaluation of the distance between the new and the old case, a proper selection of the color space employed for the image representation is crucial with respect to the effectiveness of the metric, in terms of discrimination capability. Here, inspired by the behavior of the human visual system, we propose to use the CIE<sup>2</sup>  $L^*a^*b^*$  perceptually uniform color space. In fact, at a first glance, the measure  $\Delta s$  of the difference perceived by a human observed between the stimuli produced by two uniform images satisfies the relationship

<sup>2</sup> CIE: Comit at International d’ clairage.

$$(\Delta s)^2 = (\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2$$

Then, given the  $i$ -th image pattern related to the  $i$ -th case, let

$$\mathbf{g}^{(i)}(\mathbf{x}) = \begin{bmatrix} L^{*(i)}(\mathbf{x}) \\ a^{*(i)}(\mathbf{x}) \\ b^{*(i)}(\mathbf{x}) \end{bmatrix}$$

be its representation in the CIE  $L^*a^*b^*$  color space, where  $\mathbf{x} = [x_1 \ x_2]^T \in \mathbb{R}^2$  denotes the coordinates of a point of the real plane. To build the index associated to it, we represent  $\mathbf{g}^{(i)}(\mathbf{x})$  by means of its Laguerre-Gauss transform (LGT), that constitutes the polar counter-part of the Hermite Transform proposed in [15]. Let us observe that the elements of the basis employed for the image representation are tuned to specific, visually relevant, image features. In fact, LG functions of the first order are matched to edges, while second order LGF are matched to lines and third and fourth order LGF to forks and crosses respectively. Thus, the current case base indexes are build upon the kind of features that the human visual system is supposed to use too. In addition, due to this inherent capability, the representation of  $\mathbf{g}^{(i)}(\mathbf{x})$  is usually rather sparse. But, use of the LGT is not only effective for the compactness of the associated index. One of the most relevant benefits of this representation is the availability of fast algorithms for the evaluation of the  $L^2$ -norm of the difference between the image at hand and the patterns stored in the case dictionary, irrespective of their location and orientation [12]. As detailed in appendix, given a digital image, its representation stored in the Case Library is constituted by a sparse matrix  $\mathbf{C}_i$  whose elements are obtained by shrinking the LGT coefficients given by (1).

Then, as detailed in [16], the “rotation invariant” similarity metrics between the actual image  $\mathbf{f}(\mathbf{x})$  with LGT shrunk coefficients  $\mathbf{D}_{n,k}(\mathbf{m})$  and the  $i$ -th pattern with LGT coefficients  $\mathbf{C}_{n,k}^{(i)}(\mathbf{x}_0)$ , associated with the  $L^2$ -norm is

$$\Delta(\mathbf{f}, i) = \min_{\mathbf{b}_i \in \mathbf{M}} \sum_{n=-\infty}^{+\infty} \mathbf{M}_n^{(i)}(\mathbf{b}_i) \cos \left[ n\hat{\varphi}_i - \eta_n^{(i)}(\mathbf{b}_i) \right] \quad (1)$$

where

$$M_n^{(i)}(\mathbf{b}_i) e^{j\eta_n^{(i)}(\mathbf{b}_i)} \cong \sum_k \bar{\mathbf{C}}_{n,k}^{(i)}(\tilde{\mathbf{x}}_0 + \mathbf{m}) \mathbf{D}_{n,k}(\tilde{\mathbf{x}}_0 + \mathbf{b}_i + R_{-\hat{\varphi}_i} \mathbf{m})$$

and the actual  $\hat{\varphi}_i$  image orientation is estimated as specified in ther appendix. In addition we shrink the expansion coefficients, so that:

$$C_i(n, k, \mathbf{m}) = \begin{cases} 0 & \text{if } C^{(i)}(n, k, \mathbf{m}) < \gamma \\ C^{(i)}(n, k, \mathbf{m}) & \text{otherwise} \end{cases}$$

In Figure 2 we only sketch the algorithm for the similarity metrics in pseudocode.



```

Function Distance- $\Delta(C_i, D)$  returns  $\Delta$ 
  inputs :  $C_i$ ; the  $i$ -th old case
            $D$ ; the new case
  for each  $b$  in lattice- $M$  do
    begin
      for each  $n$  in  $[0, N]$  do
        begin
           $x_n = 0$ 
          for each  $k$  in  $[0, K]$  do
             $x_n = x_n + \text{inner-product}(C_i(n, k, 0), D(n, k, b))$ 
          end
           $\hat{\varphi} \leftarrow \text{best-estimate}(x)$ 
           $R_{-\hat{\varphi}} \leftarrow \text{rotation-matrix}(\hat{\varphi})$ 
           $\Delta(f, i, b) \leftarrow 0$ 
          for each  $n$  in  $[0, N]$  do
            begin
              for each  $k$  in  $[0, K]$  do
                for each  $m$  in lattice- $M$  do
                   $\Delta(f, i, b) = \Delta(f, i, b) +$ 
                     $\text{inner-product}(C_i(n, k, m), D(n, k, b + R_{-\hat{\varphi}} m))$ 
                end
              end
            end
           $\Delta \leftarrow \min_{b \in M} \text{Re}[\Delta(f, i, b)]$ 
        end
      returns  $\Delta$ 
    end
  end

```

**Fig. 2.** The Distance Metrics Algorithm

### 3 A Case Study: Image Recognition for an Autonomous Mobile Robot

In this section we present an application of the approach proposed in respect to the problem of a robot autonomous navigation in an environment of which it has no prior knowledge [19,20,3].

The model we consider (NOMAD 200 by Nomadic Technologies) was equipped with a ring comprising 16 ultrasonic rangefinders (POLAROID), with an independent rotation ability; in order to increase measured accuracy, at each position of the environment the ring is rotated twice by an angle of  $7.5^\circ$ : as a result,  $16 \times 3 = 48$  range readings are obtained at each measure point. The working principle of ultrasonic sensors is simple: they beam a packet of ultrasonic waves, identify the resulting echo and, based on the measurement of the time elapsed from the moment of emission to the resulting echo, provide the range between the robot and the detected obstacle [21].

In addition these devices are particularly common since – besides their decidedly easy-to-use property – they guarantee a satisfactory compromise between cost and accuracy, an important aspect that cannot be underestimated, espe-

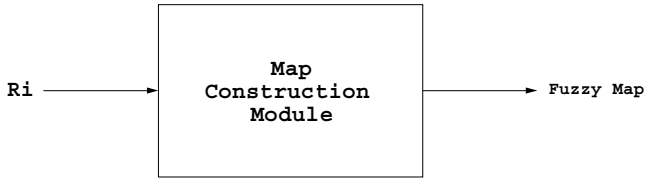


**Fig. 3.** Map Building Module

cially if one considers the idea of subsequently marketing the prototype. This is the reason behind the frequent choice of cheaper sensing devices instead of more sophisticated sensors such as laser or infrared ones, or even video cameras. Obviously this choice implies other consequences, indeed, by using ultrasonic sensors one must be prepared to face a sizeable amount of uncertainty that they entail in the measurement process [14]. In fact, in principle, the map building phase of the environment surrounding the robot could be viewed as a decision problem, which consists in establishing, for each point in the area of interest (the universal set  $U$ ), which of the complementary subsets (the empty space or the occupied space, two crisp sets giving a partition of  $U$ ) it belongs to. The significant amount of uncertainty given by an ultrasonic measurement process somewhat compromises its support of such a clear-cut separation. Therefore the map building module could be seen more efficient if developed as shown in Figure 3.

In other words, having as input the sequence of range readings  $R_i$ , collected at known sensor locations (obviously the achieved representation of the environment is less coarse if the measurement points are numerous and well distributed), and as output a fuzzy set, so-called the “fuzzy map”, which supplies information on the collision risk in respect to each point in the environment.

Previous studies in this field [19,20] have emphasized the advantages gained from the use of Fuzzy Logic techniques and, in particular, of the Fuzzy Set theory, in respect to sensor-based navigation architectures. In fact, the origin of this theory, stemming from less conservative axioms as opposed to more conventional approaches (Probability Theory and Belief Measures), makes it a flexible and thoroughly efficient tool for the development of an environment representation on the basis of uncertain data [7]. Furthermore, its contribution is particularly relevant in those cases, far from remote, where the measurement is polluted by random and irregular values (outliers) [2,25,13].

We therefore opted for an intrinsically uncertain representation of the environment thus obtaining a gray level map such as the one shown in the figure, where the membership function  $m$  (which may assume any real value within the interval  $[0,1]$ ) quantifies the possibility of each position being taken up by an obstacle; in this particular case the environment surrounding the robot was subdivided in a matrix with  $40 \times 40$  square cells, their side  $d = 0.1$  m, as shown in Figure 4 (representing a crossing to the right).

Interestingly, the result is a representation similar to an “occupancy grid” usually obtained using stochastic techniques [4]; inside this grid, the darker zones

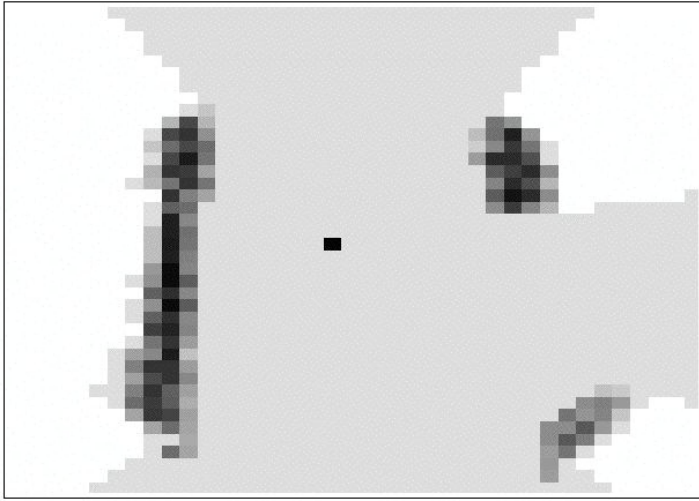


**Fig. 4.** Fuzzy Map Processed by the System

are the ones where the respective value  $m$  is higher, i.e., they reflect a greater level of membership to the fuzzy set of the dangerous cells, whereas the white areas correspond to the cells belonging to unexplored zones, i.e., for which the sensors have not provided any data.

In specific applications, as the robot mission control, used as test bed of the proposed CBR application, the description of the occupancy grid given by the LGT is still too rich, and further simplifications are possible. More in detail, the strategy selection for the short term robot guidance requires a description of the location of obstacles and constraints of the actual scenario. On the other hand, as shown in Figure 4, the low cost sensors actually employed for the acoustic imaging task, produce very rough, noisy images. As enlightened by the experimental activity, a viable approach to a sub-optimal, low complexity guidance and control system consists in replacing the full 2-D description with the polar map of the angular location of obstacles which are in the robot vicinity, i.e., whose distance is less than a predefined value [19,20]. Due to the mentioned sensor limitations, we are able to draw just a fuzzy polar map (see Figure 6). Nevertheless, we still require the ability of the CBR system to retrieve a proper strategy even when the actual situation and a case already known differ just for a rotation of the robot attitude. Since rotations produce shift in the polar map, to design an index system and its associated metric, independent from rotations and scaling we resort to a wavelet transform representation, as in the full resolution 2D situation. Once again, the properties of clustering within the same scale and persistence across different scales of the wavelet transform allows to create extremely compact indexes. In addition, when bi-orthogonal wavelets are

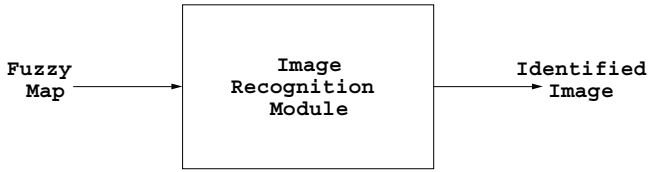


**Fig. 5.** Image Recognition Module

employed, the metric associated to the  $L^2$ -norm can be very efficiently evaluated. In addition, we expect that the differences between new and old cases will affect mainly the details of the fuzzy polar map, related to the finest scales.

Accordingly, the module for image recognition may be given as represented in Figure 5.

As far as the output is concerned, the case has been carefully assessed and deemed it absolutely not reductive to group the infinite situations that the robot could encounter during its navigation essentially into five different “objects”:

- Passage
- Corner
- Crossing
- End Passage
- Open space

thus forming a finite set.

The library of the 1-D wavelets transforms of the fuzzy polar map represents the history of the robot’s experiences gained until that moment.

## 4 Conclusions

Traditional methodologies of pattern recognition usually require the availability of templates of the objects we want to classify. This template collection reflects the a priori knowledge we have about the problem to be solved by the image classifier. However, in practical cases, like the cited robot autonomous navigation, or the browsing of large multimedia databases, the prior knowledge could be rather poor, thus leading to a rise of misclassifications. In our contribution, we encapsulated traditional feature extraction and comparison algorithms into a CBR shell that allows constant update of the environment knowledge, or, in other words, of the library of templates as well as of the library of objects to be recognized. We remark that, in principle, there is no limit to the kind and number of features that can be employed for case retrieval. Nevertheless features have to be selected according to their effectiveness in gathering the information the users is interested to. Presently an extensive test of our system is in progress. Further developments of this approach could consist in: the introduction in the features extraction module of data fusion algorithms for the combined use of

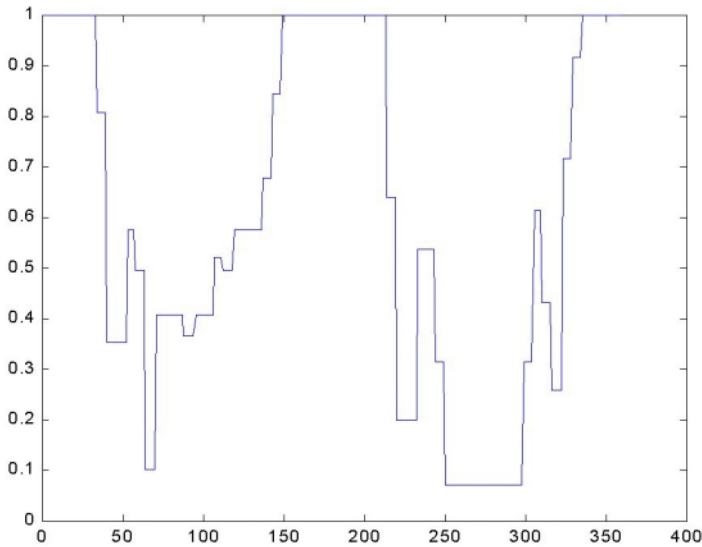


**Fig. 6.** Polar Representation of the Map of Figure 4

different features, as well as, in the refinement of the case library and object update procedure, to incorporate the evaluation of the effects produced on the environment by the current classification. For instance, in the robot guidance, the update could account for the effectiveness of the move induced by the classification.

## References

1. Androutsos, D. A., Plataniotis, K. N. and Venetsanopoulos, A. N.: Directional Detail Histogram for Content Based Image Retrieval *DSP97 13th Int. Conf. On Digital Signal Processing*, Proc. July 2-4, 1997, Santorini, pp. 225-228.
2. Bandemer, H. and Nather, W.: *Fuzzy Data Analysis*, Kluwer Academic Publishers, 1992. 448
3. Cox, I. J. and Willfong, G. T.: *Autonomous Robot Vehicles*, Springer-Verlag, 1990. 447
4. Elfes, A.: Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception. In: Iyengar, S. S., and A. Elfes (eds.) *Autonomous Mobile Robots: Perception, Mapping, and Navigation*, IEEE Computer Society Press, 1991, pp. 60-71. 448
5. Ficet-Cauchard, V., Porquet, C. and Revenu, M.: An Interactive Case-Based Reasoning System for the Development of Image Processing Applications. In: B. Smyth and P. Cunningham (eds.) *Advances in Case-Based Reasoning, Proc of EWCBR-98*, Lecture Notes In Artificial Intelligence, **1488**, Springer, 1998, pp. 437-447. 443

6. Grimnes, M. and Aamodt, A.: A Two Layer Case-Based Reasoning Architecture for Medical Image Understanding. In: I. Smith and B. Faltings (eds.) *Advances in Case-Based Reasoning, Proc of EWCBR-96*, Lecture Notes In Artificial Intelligence, **1168**, Springer, 1996, pp. 164-178. 443
7. Henkind, S. J. and Harrison, M. C.: An Analysis of Four Uncertainty Calculi. *IEEE Transactions on Systems, Man, and Cybernetics*, **18**(5), 1988, pp.700-714. 448
8. Hsu, Y. N. and Arsenault, H. H.: Optical Pattern Recognition Using Circular Hermonic Expansion. *Applied Optics*, **21**, November 1982, pp. 4016-4019. 445
9. Jacovitti, G. and Neri, A.: Multiscale Image Features Analysis with Circular Harmonic wavelets. In: *Wavelets Applications In Signal and Image Processing III*, Proc. of SPIE 2569, July 1995, pp. 363-372. 445
10. Jacovitti, G., Manca, A. and Neri, A.: Hypercomplete Circular Harmonic Pyramids. In: *Wavelets Applications In Signal and Image Processing IV*, Proc. of SPIE 2825, August 1996, pp. 352-363. 445
11. Jacovitti, G. and Neri, A.: Content based Image Classification with Circular Harmonic Wavelets. In: *Hybrid Image and Signal Processing VI*, Proc. of SPIE 3389, 1998. 445
12. Jacovitti, G. and Neri, A.: Multiscale Circular Harmonic wavelets: a Tool for Optimum Scale-Orientation Independent Pattern Recognition. In: *Wavelets Applications V*, Proc. of SPIE 3391, 1998. 445, 446, 454
13. Klir, G. J., and T. A. Folger.: *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, 1988. 448
14. Leonard, J. J. and Durrant-White, H. F.: *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers, 1992. 448
15. Martens, J. B.: Local Orientation Analysis in Images by Means of the Hermite Transform. *IEEE Transactions On Image Proc.*, **6**(8), July 1997, pp. 1103-1116. 445, 446
16. Micarelli, A. and Sciarbone, F.: A Case-Based System for Adaptive Hypermedia Navigation. In: I. Smith and B. Faltings (eds.) *Advances in Case-Based Reasoning, Proc. of EWCBR-96*, Lecture Notes in Artificial Intelligence, **1168**, Springer-Verlag, Berlin, 1996, pp. 266-279.
17. Micarelli, A., Sciarbone, F., Ambrosini, L. and Cirillo, V.: "A Case-Based Approach to User Modeling". In: B. Smyth and P. Cunningham (eds.) *Advances in Case-Based Reasoning, Proc. of EWCBR-98*, Lecture Notes in Artificial Intelligence, **1488**, Springer-Verlag, Berlin, 1998, pp. 310-321.
18. Michel, S., Karoubi, B., Bigun, J. and Corsini, S.: Orientation Radiograms for Indexing and Identification in Image Databases. *Signal Processing VIII*, Proc. of EUSIPCO, Trieste, Italy, 10-13 Sept. 1996, pp. 1693-1696. 445
19. Oriolo, G., Ulivi, G. and Vendittelli, M.: Motion Planning with Uncertainty: Navigation on Fuzzy Maps. *Proc. 4th IFAC Symposium on Robot Control*, (SY.RO.CO.94), **1**, Capri, 1994, pp. 71-78. 447, 448, 449
20. Oriolo, G., Ulivi, G. and Vendittelli, M.: On-Line Map Building and Navigation for Autonomous Mobile Robots. *Proc. 1995 IEEE Int. Conference on Robotics and Automation*, Nagoya, Japan, 1995, pp. 2900-2906. 447, 448, 449
21. Polaroid Corporation *Ultrasonic Ranging System*, 1987. 447
22. Ravichandran, G., and Trivedi, M. M.: Circular-Mellin Features for Texture Segmentation. *IEEE Transactions on Image Processing*, **4**(12), December 1995, pp. 1629-1640. 445

23. Sheng, Y. and Arsenault, H. H.: Object Detection from a Real Scene using the Correlation Peak Coordinates of Multiple Circular Harmonic Filters. *Applied Optics*, **28**(2), January 1989, p. 245. 445
24. Simoncelli, E. P.: A Rotation Invariant Pattern Signature. *Proc. of the IEEE Int. Conf. On Image Proc. ICIP 96*, Lausanne, Switzerland, September 16-19, 1996. 445
25. Zimmermann, H.-J.: *Fuzzy Set Theory and Its Applications*, Kluwer Academic Publishers, 1991. 448

## Appendix

Given an image  $\mathbf{g}^{(i)}(\mathbf{x})$  and a lattice

$$\mathbf{M} = \{\mathbf{m} = [m_1 \delta \ m_2 \delta]^T, m_1 = 1, \dots, N_1, m_2 = 1, \dots, N_2\}$$

defined on the image support, we can locally expand  $\mathbf{g}^{(i)}(\mathbf{x})$  around any lattice node, so that

$$\mathbf{g}^{(i)}(\mathbf{x}) = \frac{1}{\sum_{\mathbf{m} \in \mathbf{M}} V\left(\frac{|\mathbf{x} - \mathbf{m}|}{\sigma}\right)} \sum_{\mathbf{m} \in \mathbf{M}} \sum_n \sum_k \mathbf{C}_{n,k}^{(i)}(\mathbf{m}) \frac{1}{\sigma} \mathcal{L}_k^{(n)}\left(\frac{|\mathbf{x} - \mathbf{m}|}{\sigma}, \gamma(\mathbf{x} - \mathbf{m})\right)$$

where

$$\mathcal{L}_k^{(n)}(r, \gamma) = (-1)^k 2^{\frac{(|n|+1)}{2}} \pi^{\frac{|n|}{2}} \left[ \frac{k!}{(|n| + k)!} \right]^{\frac{1}{2}} r^{|n|} L_k^{|n|}(2\pi r^2) e^{-\pi r^2} e^{jn\gamma}$$

are the Laguerre-Gauss (LG) circular harmonic functions, and  $L_k^{(n)}(t)$  are the generalized Laguerre polynomials defined by the Rodrigues formula

$$L_k^{(n)}(t) = \frac{t^{-n} e^t d^k}{k!} \frac{d^k}{d_t^k} [t^{k+n} e^{-t}] = \sum_{h=0}^k (-1)^h \binom{n+k}{k-h} \frac{t^h}{h!}$$

The expansion coefficients are given by the inner products between the current pattern and elements of the representation basis as follows:

$$\mathbf{C}_{n,k}^{(i)}(\mathbf{m}) = \langle \mathbf{g}^{(i)}(\mathbf{x}) V\left(\frac{\mathbf{x} - \mathbf{m}}{\sigma}\right), \frac{1}{\sigma} \mathcal{L}_k^{(n)}\left(\frac{|\mathbf{x} - \mathbf{m}|}{\sigma}, \gamma(\mathbf{x} - \mathbf{m})\right) \rangle \quad (2)$$

where  $V(\mathbf{x})$  is a Gaussian weight function, i.e.:  $V(\mathbf{x}) = e^{-\pi|\mathbf{x}|^2}$ . Let  $R_{\varphi}$  be the rotation operator. Then, as detailed in [16], the distance between the actual image  $\mathbf{f}(\mathbf{x})$  with LGT coefficients  $\mathbf{D}_{n,k}(\mathbf{m})$  and the  $i$ -th pattern with LGT coefficients  $\mathbf{C}_{n,k}^{(i)}(\mathbf{x}_0)$ , associated with the  $L^2$ -norm is

$$\min_{\varphi_i, \mathbf{b}_i} \{[\mathbf{f}(\mathbf{x}) - \mathbf{g}^{(i)}(R_{\varphi_i}(\mathbf{x} - \mathbf{b}_i))]^T [\mathbf{f}(\mathbf{x}) - \mathbf{g}^{(i)}(R_{\varphi_i}(\mathbf{x} - \mathbf{b}_i))]\}$$

On the other hand,

$$[\mathbf{f}(\mathbf{x}) - \mathbf{g}^{(i)}(R_{\varphi_i}(\mathbf{x} - \mathbf{b}_i))]^T [\mathbf{f}(\mathbf{x}) - \mathbf{g}^{(i)}(R_{\varphi_i}(\mathbf{x} - \mathbf{b}_i))] = \|\mathbf{f}(\mathbf{x})\|^2 + \|\mathbf{g}^{(i)}(\mathbf{x})\|^2 - 2Re [\langle \mathbf{f}(\mathbf{x}), \mathbf{g}^{(i)}(R_{\varphi_i}(\mathbf{x} - \mathbf{b}_i)) \rangle]$$

due to the properties of the LG functions, since when  $\delta = \sigma/\sqrt{\pi}$ ,

$$\sum_{\mathbf{m} \in M} V \left( \frac{|\mathbf{x} - \mathbf{m}|}{\sigma} \right) \cong 1$$

denoting with the over bar the complex conjugate of a complex number, the inner product in the previous formula can be approximated as follows:

$$\langle \mathbf{f}(\mathbf{x}), \mathbf{g}^{(i)}(R_{\varphi_i}(\mathbf{x} - \mathbf{b}_i)) \rangle \cong \sum_{\mathbf{m} \in \mathbf{M}} \sum_k \overline{\mathbf{C}}_{n,k}^{(i)}(\tilde{\mathbf{x}}_0 + \mathbf{m}) \mathbf{D}_{n,k}(\tilde{\mathbf{x}}_0 + \mathbf{b}_i + R_{-\varphi_i} \mathbf{m})$$

where the pattern orientation is estimated by means of the recursive formula (see [12]),

$$\hat{\varphi}_i^{(m)} = \hat{\varphi}_i^{(m-1)}(\mathbf{b}_i) - \frac{\sum_{n=-\infty}^{\infty} n M_n^{(i)}(\mathbf{b}_i) \sin \left[ n \hat{\varphi}^{(m-1)}(\mathbf{b}_i) - \eta_n^{(i)}(\mathbf{b}_i) \right]}{\sum_{n=-\infty}^{\infty} n^2 M_n^{(i)}(\mathbf{b}_i) \cos \left[ n \hat{\varphi}^{(m-1)}(\mathbf{b}_i) - \eta_n^{(i)}(\mathbf{b}_i) \right]}$$

with

$$M_n^{(i)}(\mathbf{b}_i) e^{j\eta_n^{(i)}(\mathbf{b}_i)} \cong \sum_{\mathbf{m} \in \mathbf{M}} \sum_k \overline{\mathbf{C}}_{n,k}^{(i)}(\mathbf{m}) D_{n,k} \mathbf{b}_i$$



# The Life Cycle of Test Cases in a CBR System

Mirjam Minor and Alexandre Hanft

AI Lab, Dept. of Computer Science  
Humboldt University, D-10099 Berlin  
{minor,hanft}@informatik.hu-berlin.de

**Abstract.** In this article, a case-based approach for managing cases with a life cycle is introduced. The authors present an application for the accompaniment and support of software engineers in their work of specifying test cases. Some general aspects of corporate knowledge editing are discussed. The model of life cycles provides a solution for editing and retrieving cases with several degrees of maturity. It is realised by persistent case numbers, explicit revision states, and a multi-layered similarity function. Some experiments are performed with a prototypical system.

**Keywords:** Case-based reasoning, corporate memories, knowledge acquisition, maintenance.

## 1 Introduction

Managing corporate knowledge is alongside e-commerce one of the most successful areas in CBR research and practice. Often, the following basic idea has been adopted from knowledge management and realised in a CBR application: that tacit knowledge, i.e. knowledge that resides in individuals, has to be codified in symbols which other people can understand (words, pictures, sounds or actions) [20]. As CBR systems are less knowledge intensive than other AI technologies like rule-based systems, the CBR researchers turned their main attention for a long time to other problems than the life cycle of knowledge. Now solutions for some basic questions have been found, and CBR systems are able to handle large-scale and long-term case bases. Recently, the significance of knowledge acquisition, maintenance of knowledge, knowledge assessment, knowledge adaptation to mutating environments, and re-use of knowledge has increased [7,1,4,12]. In addition to methodical and psycho-social difficulties that might appear, some more technical problems have to be solved. Although the knowledge containers of case-based systems are less complex than the knowledge bases of other AI techniques, the processes concerning the knowledge are quite complex, like handling patchy and vague knowledge, managing multiple authorship of one piece of knowledge, and dealing with variability and expiration of knowledge.

In this paper, an intuitive model for knowledge editing processes is presented, namely the life cycle model for semi-structured cases. It has been implemented in a CBR system that supports the generation of test cases for functional software

testing [16]. This application is the result of a co-operation with the PSI AG, Berlin.

First, in Section 2, the management of test knowledge with CBR and some general aspects of editing knowledge containers are discussed. The basics and realisation of the life cycle model are explained in Section 3. In Section 4, some experimental results are investigated. Related work is presented in Section 5 and, finally, some conclusions are drawn in Section 6.

## 2 Managing Test Knowledge

The CBR system presented here has been designed for the support and accompaniment of software engineers in their work of specifying test cases. In all phases of developing software, knowledge about functional tests emerges. Figure 1 illustrates the different steps of producing software and test cases following the V model [18]. From the analysis of the requirements up to the real implementation phase, the engineers have to design test cases. When writing the use cases, the developers can concurrently derive drafts of test cases from the use cases. In accordance with the ISO 9001, the execution of tests is a part of the development of reliable software.

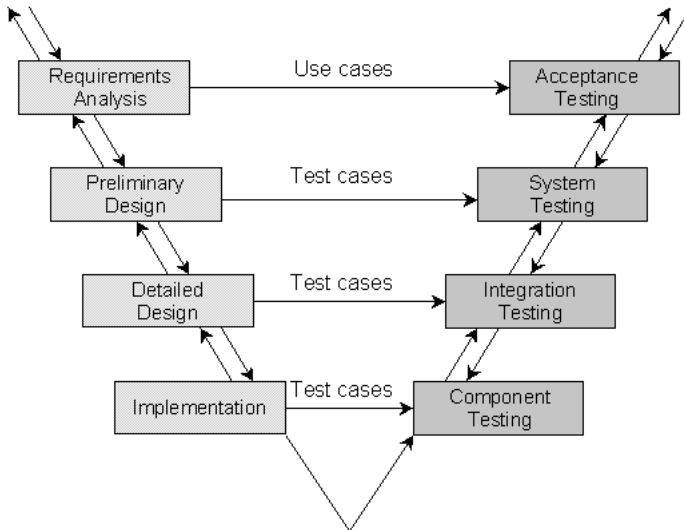


**Fig. 1.** The V Model of Software Engineering

Regardless of the chosen software developing model, the CBR system is available in every developing stage. It makes the actual test phase after the imple-

mentation easier if the users have acquired test cases already in early phases. Even retrieved ideas of tests are useful when conceiving a test specification, i.e. a suitable collection of test cases. Further advantages emerge when somebody is writing a new test case: The CBR system makes it easy to re-use ideas and good formulations either from former projects or from the present project as well as to verify whether the case base contains duplicates. Therefore, the actual written case can be used as a query, and the retrieval result presents cases that are similar to the current case.

The functionality of the case-based test-manager-tool is the same when using other developing models, for instance the standard waterfall model or the spiral model. The CBR system always collects test cases and ideas for test cases, stores them in a case base, and provides an easy access via a retrieval mechanism.

## 2.1 The Test Cases

A case contains a test case that consists basically of the following three parts:

- a textual description of the functionality of a piece of software that should be tested,
- instructions how to perform this test, and
- an expected result when performing the test.

These three parts are reflected by three textual sections in the case structure. Additionally, a case has some attribute-value pairs, e.g. for information on the concerned module.

Example of a test case:

```

<CASE_NUMBER>
122
<RETRIEVAL_ATTRIBUTES>
OBJECT = basic processing
CATEGORY = simulation
<INFORMATION_ATTRIBUTES>
GENERATED_ON = 24.12.1998
GENERATED_BY = smith
<DESCRIPTION>
The online data model manages measuring values accepting a
limit in the field "upper alarm limit".
<INSTRUCTION>
In the test station, a value higher than the limit is set.
<EXPECTED_RESULT>
The value is processed as described in chapter 5: entry stat.
limit ZA Go warning.

```

The case-based system stores such semi-structured test cases in different degrees of maturity. At the same time, the case base contains finished cases as well as fragmentary cases that have only some filled sections. The cases are still

evolving when they are stored and used in the case base. Intervals between authoring processes can last up to several months. The cases are expanded section by section and used in retrieval processes meanwhile. For instance, a fragmentary case is created on-the-fly when a software developer has a vague idea of a test object but has not yet specified how to perform a concrete test. The case is integrated into the case base and accessible for retrieval processes. It may rest for several weeks until the author or another software engineer fills out some more parts, for example, when the concerned module has been finished and shall be tested systematically. This gradual evolution of cases is the basis for the life cycle model presented in Section 3.

## 2.2 Editing the Knowledge Containers

In building and maintaining knowledge repositories the idea of having knowledge administrators has been realised in several CBR systems. The following two examples may illustrate the role that such administrators are playing: In the Simatic application [14], the indexing dictionaries are filled and maintained by experienced staff members while the case documents are written by all hotline support members. Another example is the CBR PEB project in the Web, where a maintenance team checks incoming information “if it is valuable and complete or just nonsense input from a surfer” [3, page 46]. Concentrating on a few domain experts editing the knowledge containers increases the probability that the knowledge is current, consistent, and focused.

However, in the application presented in this paper it is undesirable to edit the knowledge containers in a small, specialised group. The development of test specifications is part of the service performed for the customer, and all members of the team are responsible for it. Every user of the CBR system is a possible knowledge administrator who contributes or expands test cases, vocabulary, and ontological information. This leads to the occurrence of multiple authorship. For a brief discussion of the implications of multiple authorship, we classify knowledge containers in the sense of Richter [13, page 10 f.] into two types used in the presented project:

- weakly-structured knowledge containers (vocabulary for indexing the cases, ontological and linguistic information for providing local similarity values) and
- semi-structured knowledge containers (cases with several sections of free text or attribute-value pairs).

Containers for adaptation knowledge do not occur in the current project. At the present state of the art, the authors would rather assign adaptation knowledge containers, for instance adaptation rules, to the category semi-structured or even complex.

Depending on the type of knowledge container, it is more or less difficult to deal with multiple authorship: Modifications concerning the weakly-structured knowledge, e.g. adding a new term to the vocabulary, are quite atomic and can

be handled by a simple transaction management: Storing single modifications immediately avoids conflicts with multiple authorship. To know the author of such small pieces of knowledge is not significant at all. For instance, a particular technical term belongs normally to the corporate knowledge of a team; and it's integration to the vocabulary should not be a subject of great controversy. But editing the cases is a little bit more complicated. A database's transaction management is not enough to perform all tasks, because the good performance of the retrieval processes requires a time-consuming precompilation of the case data that is usually done off-line. On-line, several authoring and retrieval processes are interlaced. While one author is editing a case, another one should be able to use it within a retrieval process. Time displacements between the case developing processes make it difficult to provide consistent retrieval processes at any time. So, we organisationally distinguish the case data used for retrieval processes from the highly up-to-date case data for case authoring processes. Between two complete updates, newly generated test cases are stored separately. New cases are only visible from the perspective of case authoring. When an already existing case is modified, a new revision is stored that is only accessible in the event of case authoring. So, all recently edited cases are shown in two different views to retrieval and editing processes. Other revisions saved meanwhile are only stored in a history management which is not used yet. The unification of the two views either happens when an update is started explicitly or at regular intervals automatically.

### 3 The Life Cycle Model

We call the development of a case from a fragment to a complete test case *life cycle*. This term expresses intuitively what happens with a case when it is “born” as a member of the case base, resumed and amended several times, and possibly rejected sometime. In contrast to other CBR approaches [8,10], the life cycle refers not to the whole system but to a single knowledge container, namely a case. The life cycle model of cases is realised by the following issues:

- revision states of cases,
- persistent cases with unique identification numbers, and
- an appropriate similarity function that is quite robust against expansion of cases.

#### 3.1 Revision States of Cases

In it's curriculum vitae, a case passes several *revision states*. Number and order of modifications is unconstrained. A typical life cycle of a test case is shown in Figure 2. Internally, the revision state of a case is represented by values of a few organisational attributes: the developing state of the case as linguistic value (e.g. “idea for a test”) contributes to the retrieval results, while the revision number that is continuously incremented with every save process is only information

for the user. Date and author of the last update are remembered as well in organisational attributes. So, the position of a case within it's life cycle is always comprehensible.

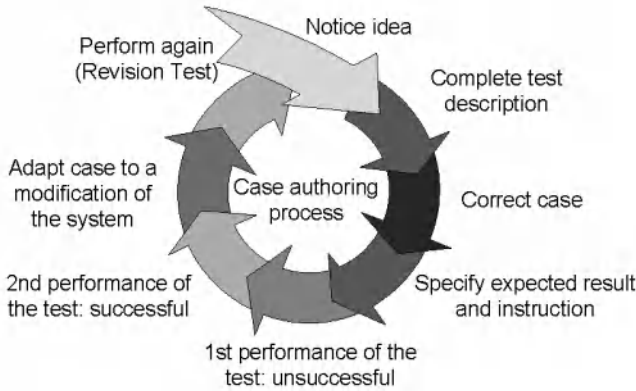


**Fig. 2.** A Typical Life Cycle of a Case

### 3.2 Persistent Cases

The concept of a life cycle requires persistent case objects with persistent identification numbers. Instead of the generation of a new case object, the case keeps the same number when it has been taken out of the case base for editing and pushed back afterwards; only the revision number increases. Even if test cases of different projects are managed in one case base, the case numbers are assigned uniquely. This has been motivated by the experience that software engineers usually remember the identification numbers of those cases they have written or edited themselves. Additionally, the membership of projects are given as attribute values for clustering the cases. Deleting cases may lead to gaps in the continuous case numbers. But it is more important to have life-long case numbers that help to easily recognise a case after diverse authoring processes.

### 3.3 Robust Similarity

Nuts and bolts of the life cycle model is a similarity function dealing with such a hybrid case base: with small and long cases, with fragmentary and finished ones, and with cases in all revision states between. The aim to manage cases in all developing states leads to a modification of the usual concept of similarity. In some publications, similarity has been replaced by usefulness [6]. Adopting this, useful cases for writing test cases are those with similar sections to a query. But both query and case, can be patchy and contain empty sections. The role of the empty sections disturbs the unity of intuitive similarity and usefulness.

When a query has filled only some of the sections, completely specified cases shall be found as well as fragments of cases in an earlier development state. On the other hand, when a query has a lot of filled sections, also cases at the beginning of their life-cycle should be considered. The similarity function relies on an optimistic view of the future, it expects that empty sections will be filled with matching contents. So, a case can be very useful to a query even if a few sections are empty in one or in both.

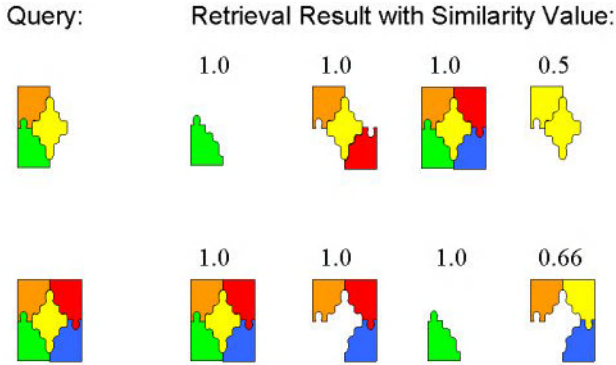


**Fig. 3.** Example of a Retrieval Result

To meet these requirements, we chose a three layer similarity function. *Partial similarity* values between corresponding sections filled in both the query and the case, are computed by means of *local similarity* values a priori gained from atomic similarities in the real world. As most of the sections of a case or a query contain textual descriptions, a method of textual case-based reasoning developed in the CBR-Answers project [15] has been applied for the partial similarity function. Broadly speaking, that means a representation of the texts by means of dictionaries of terms and comparison of the texts by means of dictionaries of synonyms and further local similarity relationships between terms. A *global similarity* function comparing a query with a case combines the partial similarity values:

$$\begin{aligned}
 \text{sim}(\text{query}, \text{case}) &= \frac{1}{\alpha} \text{sim}(S_{\text{query}}^1, S_{\text{case}}^1) + \\
 &\quad \frac{1}{\alpha} \text{sim}(S_{\text{query}}^2, S_{\text{case}}^2) + \\
 &\quad \dots + \\
 &\quad \frac{1}{\alpha} \text{sim}(S_{\text{query}}^k, S_{\text{case}}^k).
 \end{aligned}$$

If a section  $S_i$  is empty in the query or in the case, then  $\text{sim}(S_{\text{query}}^i, S_{\text{case}}^i)$  is set to 0. To reach similarity values in the range between 0 and 1, the similarity

function is scaled by  $\frac{1}{\alpha}$  where  $\alpha$  is the number of representative terms in the sections of the query that have a filled partner section in the case. The scaling can even lead to a lowering of the similarity value of a case when a query is extended and asked again. With respect to many overlaps of test cases, this effect could be useful.

## 4 Experimental Results

To show empirically that the similarity function realises the specified functionality, we implemented it in a *Case Retrieval Net* in C++. Generating the *information entities* (IEs) for the IE nodes, we copied the entries of the term dictionaries according to the number of sections. So the sections have separated sets of IEs indexing them. The partial similarity values are collected separately and then combined to give the overall similarity value. The graphical user interface is implemented as a Java Applet.

**Experimental Setup:** We used a case base with about 1400 complete test cases from one software project for gas transportation systems. We stretched this set with about 150 fragmentary cases derived from the original set by omitting sections or parts of sections. So we generated some artificial cases of early development phases that contain only a vague idea of a test. Vocabulary for indexing the textual sections and local similarity values is made up of some transferred from former projects [11], some extracted from domain-specific electronic texts.

**Method:** An evaluation of the partial similarity function that is borrowed from former textual CBR applications is described in [15]. In the current paper, we would like to show it is plausible that the multilayered similarity function behaves robustly within the life cycle of a case. We took 30 complete test cases out of the experimental case base described above. From each chosen case we derived four queries by using the entire case as the original query and then successively omitting sections to produce three variations of the query. So, we had fragmentary and complete cases within the case base as well as fragmentary and complete queries. We started the 120 retrieval processes and investigated the 30 similarity values of one particularly *well matching* non identical reference case in the case base per complete query compared with the three times 30 values between the reference cases and the three shortened queries each.

**Results:** The results of those experimental retrieval processes are shown in Figure 4. The similarity values of the particular four variations of the query and their reference case are visualised in percentage terms. As the small distances between the four lines concerning one query show, the similarity values did not differ much except for Query 25 where the four-section query had only a quite bad reference case with 62.5% similarity, but an excellent value of 100% for the one-section query. The exact values can be found in Table 1. An average retrieval process on a SUN Ultra-2 took a time of 34932.47 $\mu$ s.

**Discussion:** The performed tests make it plausible that the similarity function is robust in the desired sense, namely when a user adds or removes parts of a case. The test method simulates the real situation of users at different times



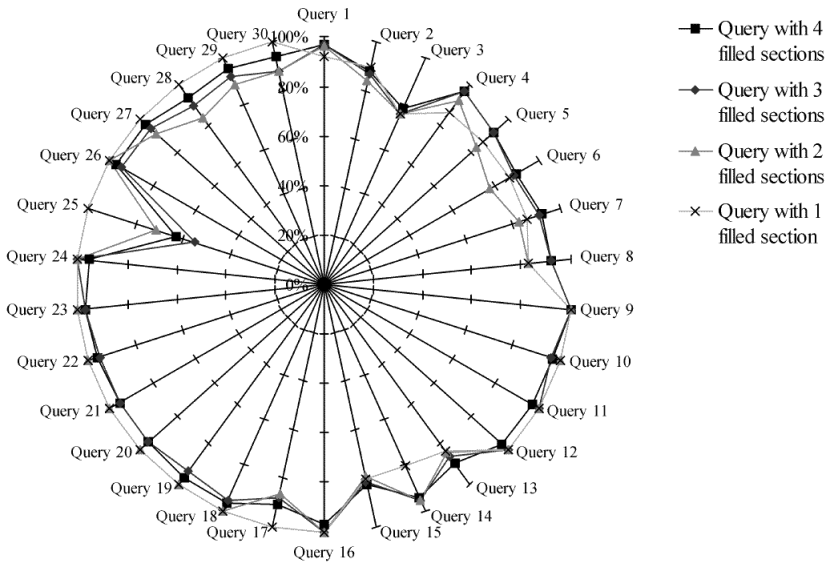


**Fig. 4.** Similarity values between the four variations of a query and the particular reference case

of the software developing process who use possible cases in various developing stages as queries for a hybrid case base. The same query in decreasing degrees of maturity kept the good matching cases in it's retrieval result with stable similarity values. That means that a user would reach the cases that are matching in the highest maturity of the query also in lower states of the query, i.e. independently of whether the conception of the query is vague or concrete. We suppose that cases that are very similar in one section also have a high coverage in the other sections. For instance, when the description of the case only differs in details like an upper and lower limit to be stepped over in a test case, the instructions how to use the test station are nearly the same.

## 5 Related Work

Another approach to manage knowledge about software engineering within a CBR system has been developed in the **INRECA** methodology [5]. In particular, software process models from CBR development are stored in three levels of abstraction in an experience base. The approach discussed in this paper is limited to only a small part of a company's knowledge about developing software, namely the knowledge concerning functional software testing. Instead, we abstain from different abstraction levels and complex case structures. Cases are written in natural language and can be directly compiled to test specifications presented to the customer.

**Table 1.** Similarity Values of Experimental Retrieval Results

| Query no. | Compared to case no. | Sim. value with 1 filled section | Sim. value with 2 filled sections | Sim. value with 3 filled sections | Sim. value with 4 filled sections |
|-----------|----------------------|----------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|
| 1         | 1010                 | 0.928571                         | 0.962963                          | 0.969697                          | 0.971429                          |
| 2         | 3                    | 0.900000                         | 0.846154                          | 0.875000                          | 0.882353                          |
| 3         | 1501                 | 0.750000                         | 0.750000                          | 0.750000                          | 0.785714                          |
| 4         | 26                   | 0.857143                         | 0.916667                          | 0.958333                          | 0.961538                          |
| 5         | 34                   | 0.857143                         | 0.833333                          | 0.916667                          | 0.923077                          |
| 6         | 32                   | 0.857143                         | 0.769231                          | 0.880000                          | 0.888889                          |
| 7         | 33                   | 0.857143                         | 0.818182                          | 0.913043                          | 0.920000                          |
| 8         | 37                   | 0.833333                         | 0.833333                          | 0.916667                          | 0.923077                          |
| 9         | 39                   | 1.000000                         | 1.000000                          | 1.000000                          | 1.000000                          |
| 10        | 9                    | 1.000000                         | 1.000000                          | 0.958333                          | 0.972222                          |
| 11        | 8                    | 1.000000                         | 1.000000                          | 1.000000                          | 0.972222                          |
| 12        | 28                   | 1.000000                         | 1.000000                          | 1.000000                          | 0.961538                          |
| 13        | 37                   | 0.833333                         | 0.842105                          | 0.857143                          | 0.888889                          |
| 14        | 44                   | 0.800000                         | 0.947368                          | 0.952381                          | 0.941176                          |
| 15        | 39                   | 0.800000                         | 0.789474                          | 0.809524                          | 0.823529                          |
| 16        | 71                   | 1.000000                         | 1.000000                          | 1.000000                          | 0.969697                          |
| 17        | 39                   | 1.000000                         | 0.857143                          | 0.875000                          | 0.909091                          |
| 18        | 8                    | 1.000000                         | 1.000000                          | 0.950000                          | 0.971426                          |
| 19        | 14                   | 1.000000                         | 1.000000                          | 0.928571                          | 0.962963                          |
| 20        | 1403                 | 1.000000                         | 1.000000                          | 0.952381                          | 0.952381                          |
| 21        | 1401                 | 1.000000                         | 1.000000                          | 0.952381                          | 0.952381                          |
| 22        | 30                   | 1.000000                         | 1.000000                          | 0.952381                          | 0.961538                          |
| 23        | 1417                 | 1.000000                         | 1.000000                          | 0.966667                          | 0.966667                          |
| 24        | 1411                 | 1.000000                         | 1.000000                          | 1.000000                          | 0.952381                          |
| 25        | 23                   | 1.000000                         | 0.714286                          | 0.545455                          | 0.625000                          |
| 26        | 69                   | 1.000000                         | 0.909091                          | 0.944444                          | 0.969697                          |
| 27        | 64                   | 1.000000                         | 0.909091                          | 0.944444                          | 0.969697                          |
| 28        | 23                   | 1.000000                         | 0.833333                          | 0.894737                          | 0.933333                          |
| 29        | 36                   | 1.000000                         | 0.875000                          | 0.923077                          | 0.954545                          |
| 30        | 1401                 | 1.000000                         | 0.875000                          | 0.875000                          | 0.941167                          |

As mentioned above, the textual similarity function developed in the **CBR-Answers** project [15] has influenced the current similarity function. In contrast to CBR-Answers, the aim of this project is to start using the case base already with very few cases and to interlace retrieval and editing processes. This includes modifying cases that are already in use as the life cycle model requires. Furthermore, the ignoring of empty sections in the scaling of the similarity values is different to CBR-Answers.

## 6 Discussion and Conclusion

In this paper, we presented a life cycle model for cases applied to the application area of generating test cases. The presented CBR system accompanies

the software engineers and collects testing knowledge in all phases of the developing process. Fragmentary and complete test cases are accessible via retrieval processes and can be enhanced several times.

The test case base is a corporate memory with multiple authorship. For dealing with the time displacements of editing processes and the interlacement of retrieval and maintenance processes, the life cycle model for cases has been developed. It is realised by persistent case numbers, explicit revision states and a multilevel similarity function that is quite robust to updates of parts of the cases.

Using a case-based system for test specification has the following benefits: Testing knowledge gained once in the development process can be conserved and accessed in the case base. Vague ideas can be inserted as a new case while on the job. Maintenance processes for already existing cases are constantly stimulated by development steps like the completion of a module. The case base is steadily growing. This will shorten the actual testing phase after the implementation. Furthermore, a retrieval result can help avoiding duplicates or finding good formulations and scenarios for an actual edited case. Specifying test cases becomes easier when good patterns from the actual or previous projects are available.

## Acknowledgements

The authors want to thank the PSI AG, Berlin, for making a practical application area available and funding this research.

## References

1. D. W. Aha and H. Muñoz-Avila. Exploring Synergies of Knowledge Management and Case-Based Reasoning. Technical Report AIC-99-008, Naval Research Lab, 1999. 455, 466
2. K.-D. Althoff, R. Bergmann, and L. K. Branting, editors. *Case-Based Reasoning Research and Development, Proc. ICCBR-99*, Lecture Notes in Artificial Intelligence, 1650. Springer Verlag, 1999. 466
3. K.-D. Althoff, M. Nick, and C. Tautz. CBR-PEB: An Application Implementing Reuse Concepts of the Experience Factory for the Transfer of CBR System Know-How. In E. Melis, editor, *CBR-PEB: An Application Implementing Reuse Concepts of the Experience Factory for the Transfer of CBR System Know-How*, pages 39–48, Saarbrücken, 1999. Universität Saarbrücken. 458
4. B. Bartsch-Spörl. Cases as Knowledge Assets. In S. Schmitt and I. Vollrath, editors, *Challenges for Case-Based Reasoning – Proceedings of the ICCBR'99 Workshops* –, pages I-25–I-28, Kaiserslautern, 1999. Universität Kaiserslautern. 455
5. R. Bergmann, S. Breen, E. Fayol, M. Göker, M. Manago, S. Schmitt, J. Schumacher, A. Stahl, S. Weiß, and W. Wilke. Collecting experience on the systematic development of cbr applications using the inreca methodology. In Smyth and Cunningham [19], pages 460–470. 463
6. H.-D. Burkhard. Extending some Concepts of CBR – Foundations of Case Retrieval Nets. In *Case-Based Reasoning Technology — From Foundations to Applications* [13], pages 17–50. 460

7. C. Carrick and Q. Yang. Activating cbr systems through autonomous information gathering. In Althoff et al. [2], pages 74–88. 455
8. T. Dingsøyr. A Lifecycle Process for Experience Databases. In S. Schmitt and I. Vollrath, editors, *Challenges for Case-Based Reasoning – Proceedings of the ICCBR'99 Workshops* –, pages I–9–I–13, Kaiserslautern, 1999. Universität Kaiserslautern. 459
9. L. Gierl and M. Lenz, editors. *6th German Workshop on CBR*, IMIB Series Vol. 7, Rostock, 1998. Universität Rostock. 466
10. M. Gómez-Albarrán, P. A. Gonzáles-Calero, B. Díaz-Agudo, and C. Fernández-Conde. Modelling the CBR Life Cycle Using Description Logics. In Althoff et al. [2], pages 147–161. 459
11. M. Kunze and A. Hübner. CBR on Semi-structured Documents: The Experience-Book and the FAI/Q Project. In Gierl and Lenz [9], pages 77–85. 462
12. D. B. Leake and D. Wilson. Combining CBR with Interactive Knowledge Acquisition, Manipulation and Reuse. In Althoff et al. [2], pages 203–217. 455
13. M. Lenz, H.-D. Burkhard, B. Bartsch-Spörl, and S. Wess. *Case-Based Reasoning Technology — From Foundations to Applications*. LNAI 1400. Springer Verlag, Berlin, 1998. 458, 465, 466
14. M. Lenz, K.-H. Busch, A. Hübner, and S. Wess. The SIMATIC Knowledge Manager. In KM and CBR [1], pages 40–45. 458
15. M. Lenz, A. Hübner, and M. Kunze. Textual CBR. In *Case-Based Reasoning Technology — From Foundations to Applications* [13]. 461, 462, 464
16. M. Minor. Managing Test Specifications with Case-Based Reasoning. In E. Melis, editor, *7th German Workshop on CBR*, pages 132–139, Saarbrücken, 1999. Universität Saarbrücken. 456
17. M. Minor and A. Hanft. Cases with a Life-Cycle. In S. Schmitt and I. Vollrath, editors, *Challenges for Case-Based Reasoning – Proceedings of the ICCBR'99 Workshops* –, pages I–3–I–8, Kaiserslautern, 1999. Universität Kaiserslautern.
18. B.-U. Pagel and H.-W. Six. *Software Engineering: Die Phasen der Softwareentwicklung*. Addison-Wesley, Bonn, erste edition, 1994. 456
19. B. Smyth and P. Cunningham, editors. *Advances in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, 1488. Springer Verlag, 1998. 465
20. O. van Ewyk. Preparing for the Knowledge Era.  
See <http://www.hci.com.au/hcisite/articles/prepare.htm>, 1998. 455

# Evaluating a Multi-modal Reasoning System in Diabetes Care

Stefania Montani<sup>1</sup>, Riccardo Bellazzi<sup>1</sup>, Luigi Portinale<sup>2</sup>, and Mario Stefanelli<sup>1</sup>

<sup>1</sup> Dipartimento di Informatica e Sistemistica  
Università di Pavia, Pavia (Italy)

<sup>2</sup> Dipartimento di Scienze e Tecnologie Avanzate,  
Università del Piemonte Orientale “A. Avogadro”, Alessandria, (Italy)

**Abstract.** In the context of Insulin Dependent Diabetes Mellitus care, we developed a decision support system that relies on a tight integration of Case Based Reasoning and Rule Based Reasoning methodologies. In this paper, we aim at presenting the evaluation strategy we have defined to test the system accuracy, safety and reliability, and the first results obtained both on simulated and on real patients data. Reliability was positively judged by a group of expert diabetologists; an increase in the performances of the system is foreseen as new knowledge will be acquired, through its usage in clinical practice.

## 1 Introduction

An effective management and exploitation of knowledge is a key requisite in the medical domain, where the introduction of Hospital Information Systems (HIS) into clinical practice has led to the necessity of keeping, distributing and reusing a large quantity of data. A proper Knowledge Management (KM) approach is particularly needed in the field of chronic diseases, among which Insulin Dependent Diabetes Mellitus (IDDM) care. IDDM patients suffer from a reduced functionality of the pancreatic beta cells, and need to inject themselves exogenous insulin 3 to 4 times a day to regulate blood glucose metabolism. Such an intensive therapy may lead to hypoglycemic episodes: Blood Glucose Level (BGL) has therefore to be frequently tested and logged. Because IDDM is a life-long condition, the amount of self-monitoring data is huge, and needs to be correctly interpreted by physicians. In particular, patients are normally visited every 2-4 months, to assess their health status, and to eventually revise therapy. The DCCT study [1] has clearly shown that the definition and realization of an appropriate individual therapy, derived from the current metabolic behavior and customized on the single patient's needs, is the key to an effective diabetes care. Therefore it would be extremely useful to increase the contacts with the diabetologist, and to remind her/him of past situations similar to the current one, belonging to the same patient or to a different person, in order to provide a context into which the present problems can be interpreted and faced. The Case Based Reasoning (CBR) methodology seems to be well suited for performing this task.

As a matter of fact, CBR promises to be a valuable way for managing *implicit* knowledge, i.e. past cases, individual expertise of workers and organizational practices [2,3]. In the IDDM domain, viewing a periodical visit as a case, a large case library can be made available. Keeping track of the problem/solution (i.e. metabolic behavior/therapy) patterns over time, it is possible to maintain expert physicians' know-how, in order to provide a good quality of care even if they move or retire. Moreover, the use of CBR techniques allows comparisons among different therapies and different physician approaches. Finally, CBR can be useful in providing a significant start point for extremely difficult cases.

Nevertheless, implicit knowledge is not the only useful knowledge *type* in the IDDM context: well-established and formalized domain knowledge is available as well, and it can be represented through different Artificial Intelligence (AI) formalisms, such as rules. Rule Based Reasoning (RBR), in fact, can be very helpful for supporting *explicit* knowledge exploitation.

To take advantage of these heterogeneous knowledge sources, we have developed a system in which CBR and RBR are integrated in a very tight way [4,5], to provide physicians with a mean for exploiting the available know-how, and with a decision support tool in therapy planning. Our solution aims at overcoming the two paradigms limitations. CBR is used to specialize and dynamically adapt the rules on the basis of the patient's characteristics and of the accumulated experience, thus avoiding the qualification problem [6]. On the other hand, if a particular patient condition is not sufficiently covered by cases, the use of rules may be exploited to learn a suitable therapy. The new case is then kept, in order to improve the competence of the Case Based component.

This Multi Modal Reasoning (MMR) system has been implemented within the T-IDDM project [7], and can be accessed through its web-based environment. The system is currently used at the Pediatric Department of Policlinico S. Matteo Hospital in Pavia.

Implementation details are described elsewhere [4,5], and are briefly recalled in section 2. The methodological issues addressed to design the system evaluation, together with the study results, are presented in section 3.

## 2 System Implementation

In defining the internal structure of cases, we adopted a classical approach, in which a case is described as a triple composed by (i) a set of feature/value pairs, (ii) a solution and (iii) an outcome: in the IDDM domain the case features are the data collected during the visit, the solution is the therapeutic protocol assigned after the features evaluation, and the outcome of such therapy is given by the number of hypoglycemic episodes and by the value of glycated hemoglobin at the following visit.

A case library of past visits has been built, and is automatically upgraded every time a patient undergoes a periodical examination. The case library structure mirrors a taxonomy of prototypical classes, that express typical problems in the age of infancy and puberty [8,9] (see figure 1). Each case belongs to a

leaf in the taxonomy tree. Case Based retrieval is therefore implemented in two steps: classification of the input case, and retrieval of past cases belonging to the identified class.

Classification and retrieval perform what is called the “situation assessment” step by using CBR terminology, and clarify the relevant context to work with, by making explicit the implicit knowledge embedded in the cases stored in the memory.

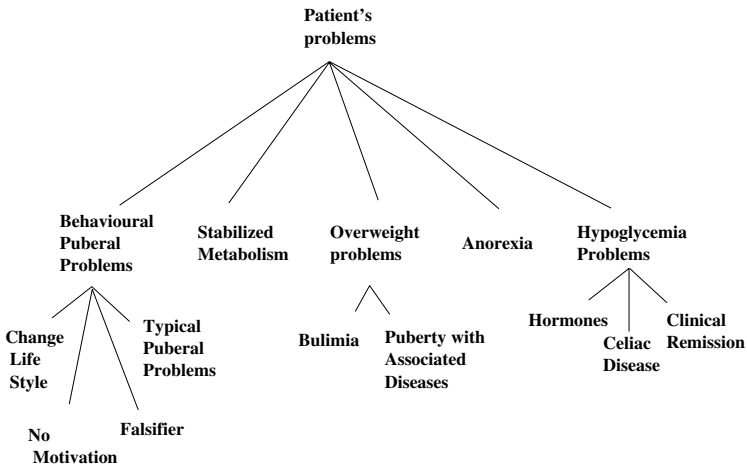


**Fig. 1.** Prototypical situations experienced by pediatric diabetic patients

Classification relies on a Naive Bayes strategy, a method that assumes conditional independence among the features given a certain class, but that is known to be robust in a variety of situations [10,11], even in the presence of conditional dependencies. Prior probabilities have been derived from medical knowledge, while posterior probabilities have been learnt from the available case library (147 real cases), using a standard Bayesian updating approach [12]. The most similar cases belonging to the most probable class(es) found in the classification step are retrieved resorting to Nearest Neighbor techniques [8,9]. Distances are calculated exploiting classical metrics, able to treat numeric and symbolic variables, and to cope with the problem of missing data [13]. When dealing with a large case base, the system performs a non exhaustive search procedure by applying an anytime algorithm called Pivoting-Based Retrieval (PBR) [14]. The idea is to restrict the case search space to the relevant part of the library, by means of the Bayesian classification step, and then to further reduce search through bound computation on the distance of stored cases with respect to the target one.

As anticipated in the previous section, the established domain knowledge is maintained in a set of production rule classes; RBR provides the possibility of

identifying the eventual metabolic alterations experienced by the patient, and of generating a therapeutic suggestion able to cope with them.

In more detail, RBR performs a sequence of reasoning tasks, each one obtained by firing the rules in the following rule classes:

- **problem identification rules.** In order to temporally contextualize the large amount of time-stamped data collected by patients during home monitoring, we have subdivided the day into seven non-overlapping time-slices, centered on the injection and/or meal times. The raw data can then be abstracted through a Temporal Abstractions (TA) technique [15]: in particular, STATE abstractions (e.g. low, normal, high values) are extracted and aggregated into intervals called *episodes*. From the most relevant episodes, it is possible to derive the BGL *modal day* [15], an indicator able to summarize the average response of the patient to a certain therapy. When the frequency (called *minimum probability*) of a certain BGL abstraction is higher than the  $\alpha$  threshold, and when the number of missing data (called *ignorance*) is sufficiently small to rely on such information (i.e. it is smaller than the  $\beta$  threshold), a *problem* is detected. For example, the following rule detects a hypoglycemia problem in a generic time-slice Y using the information contained in the relative modal day component X:

```

IF X IS A BGL-MODAL-DAY-COMPONENT
  AND THE TIME-SLICE OF X IS Y
  AND THE BGL-LEVEL OF X IS LOW
  AND THE MINIMUM-PROBABILITY OF X >= alpha
  AND THE IGNORANCE OF X <= beta
THEN GENERATE-PROBLEM HYPOGLYCEMIA AT Y

```

where the  $\alpha$  and  $\beta$  default values, derived from medical knowledge, are equal to 0.3 and 0.8 respectively;

- **suggestion generation rules.** For each detected problem, a set of alternative suggestions on insulin administration, to be applied to the therapeutic protocol, is produced;

- **suggestion selection rules.** The most suitable and effective suggestions are chosen among all the available ones;

- **protocol revision rules.** Finally, the current protocol is adjusted on the basis of the selected suggestions, and is listed with other library protocols that would fit the current patient's condition.

In the MMR methodology, the **problem identification** and **suggestion generation rule classes** are affected by the integration procedure. Integration takes place within the problem solving cycle, in the following way:

- the input case is classified;
- the most probable class information is used to specialize the **problem identification rules** parameters. For example, in the hypoglycemia detection rule described above, when dealing with patients suffering from anorexia,  $\alpha$  has to be set to 0.2, and  $\beta$  to 1 [4];
- the retrieval step is performed;



- some simple statistics are calculated on the retrieved cases, to set the **suggestion generation rules** parameters (i.e. the number of insulin injections in a day, the variation in the daily insulin requirement, and the variation of a single insulin dose);
- the RBR cycle is hence normally completed, by firing the suggestion selection and the protocol revision rules.

As the case library grows, the MMR tool is able to perform decision support at various levels of complexity: first, when the stored information is of poor quality (i.e. it presents significant gaps in the knowledge space domain), RBR can be applied to propose a therapy, without resorting to CBR. The new case is then kept in the memory in order to learn from past experience a possible way for filling a competence gap. As far as new information becomes available in the case library, making its competence grow, retrieval results become more reliable. Rule parameters are then tuned on the basis of Case Based retrieval results; the therapeutic advice is dynamically adapted to the proper interpretation context and to the single patient's needs. In other words, general rules are tailored on specific patients, by refining the suggested therapy in an appropriate way.

### 3 System Evaluation

Before implementing the integrated methodology into a real clinical setting, it has been necessary to evaluate some key factors of the system, i.e. the CBR classifier *accuracy*, the MMR suggestions *safety* on real and simulated cases, and the MMR suggestions *reliability*, through a comparison with experts opinions.

As previously noted, the usefulness of retrieval results is strongly influenced by the case library content, because the presence of competence gaps may lead to wrong or unfit rule specialization. RBR can hence be viewed as an application of MMR, when the implicit information is of poor quality. Therefore, to take into account the key role of the case library content, in our evaluation methodology we compared MMR suggestions with RBR ones.

#### 3.1 Classification Accuracy

The Bayesian classifier devoted to the context identification task has been tested through a leave-one-out cross validation technique [9], applied to a case library composed by 147 real cases, extracted from the histories of 29 pediatric patients. Moreover, we asked a diabetologist other than those who provided the knowledge to judge how the system was able to classify 14 new real patients' cases.

The classifier performed a correct classification on the 83% of the whole case base. In the 98% of the examples the correct class was one of the two most probable classes [6]. The expert physician who provided a judgment on the classifier soundness applied to the 14 real patients cases confirmed the overall positive outcome.

### 3.2 MMR Safety

RBR and MMR suggestions have been compared in two fashions:

- *retrospectively*: by running the two systems independently on 37 cases, belonging to 6 real pediatric patients, and by providing a qualitative judgment on the soundness of the answers;
- *prospectively*: by running the two systems on simulated cases. Through a realistic diabetic patient simulator [16], we were able to reproduce the behavior of pediatric patients experiencing the problems described by the taxonomy. The systems performances were judged in terms of number of therapy adjustments required to reach normoglycemia, simulating new BGL data after each revision. Safety results were satisfactory in both the evaluation procedures:
- *retrospective evaluation*: in 55% of the 37 real cases MMR suggested more substantial changes to the insulin administration in comparison to RBR. In this way, it was more prompt to react against hypoglycemia or hyperglycemia, depending on what was the most relevant problem at hand. In an additional 40% MMR and RBR outcomes were comparable, and suitable to the input situation. In the remaining 5%, the CBR tool succeeded in retrieving only one past case, unfit for the examples under examination. This is the typical situation due to the presence of a competence gap: a larger case library could increase reliability (see section 3.4). Nevertheless, as previously noted, every time the retrieval information is poor our system applies RBR without integration;
- *prospective evaluation*: all the tests on simulated data confirmed the ability of MMR to suggest sharper changes in insulin therapy. A lower number of adjustment was hence required to stabilize the sample patient when adopting MMR in comparison to RBR [4,5,17].

### 3.3 MMR Reliability

In several clinical domains, especially when dealing with therapy definition, experts may disagree on how to proceed in a certain situation. Moreover, it may be unrealistic to determine a gold standard to be followed through a sufficiently general clinical trial [18]. Diabetes therapy belongs to this category. In absence of a therapeutic behavior that is right in an absolute sense, a decision support system *correctness* can not be assessed. Only *reliability* may be addressed, making a comparison between the system's advice and the opinion of a group of experts.

Our methodology for reliability evaluation consisted of the following two steps:

- 1 - *therapy advice generation*: two independent specialists, not familiar with our tool, were asked to formulate a therapy for 30 of the 37 real patient cases already used for safety assessment. Their answers were indicated as A and B. MMR and RBR therapeutic suggestions were called C and D respectively;
- 2 - *peer review*: two other experts (E1 and E2), again unfamiliar with the tool, were asked to make a fully-crossed blind review of the A, B, C and D answers.

Each therapeutic advice, produced at step 1, was assigned one of the following scores: (1) ideal; (2) acceptable; (3) weakly acceptable; (4) unacceptable.

A total of 119 judgments were formulated by each peer reviewer (A provided only 29 answers, while B, C and D provided 30 each).

The results have been expressed as 4x4 tables, one for every expert, whose columns report the first level therapy advisors (A, B, C, D) and whose rows report the four qualitative judgments (1, 2, 3, 4) completed at step 2.

Two tests were selected to assess whether there were dependencies between therapy advisors and the judgment they obtained from each of the peer reviewers. First,  $X^2$  calculated as in (1) [19], was compared with  $\chi^2$  distribution with 9 degrees of freedom.

$$X^2 = \sum_i \sum_j \frac{f_{ij} - e_{ij}}{e_{ij}} \quad (1)$$

where

$$e_{ij} = \frac{f_{i0}f_{0j}}{f_{00}}$$

In formula (1),  $f_{i0}$  is the sum of all the elements in row  $i$ , in a  $I \times J$  table;  $f_{0j}$  is the sum of the elements in column  $j$ , and  $f_{00}$  is the sum of all the cells.

Additionally, we applied Goodman and Kruskal's  $\tau$  measures [19].  $\tau_b$  in (2) involves a comparison of the following two situations: we are asked to guess what is the score of a peer review judgment (i) without any further information, or (ii) knowing whether the author of the therapeutic advice at step 1 was A, B, C or D. If the two answers are unrelated, knowing the identity of the first level expert (physician or system) does not provide any additional indication on the score. Otherwise, there is an association between the categories on columns and on rows in the table.  $\tau_a$  quantifies the same concept, switching the roles of scores and first level experts. For both the statistics, 0 means total independence, while 1 stays for a strong association.

$$\tau_b = \frac{f_{00} \sum_i \sum_j \frac{(f_{ij} - e_{ij})^2}{f_{i0}}}{f_{00}^2 - \sum_j f_{0j}^2} \quad (2)$$

The results of the experts blind review on the 30 real patients cases are reported in tables 1 and 2. Table 3 is a summary table, obtained by summing tables 1 and 2 cell by cell.

From the outcome of the statistical tests  $X^2$ ,  $\tau_a$  and  $\tau_b$ , we can conclude that there is no evidence of a dependency between rows and columns: knowing the identity of the therapy advisor does not provide preliminary information on the score she/he obtained by peer reviewers, and vice-versa. In fact, the hypothesis

**Table 1.** Experts E1’s judgments.  $X^2=5.11, \tau_a=0.0148, \tau_b=0.0143$

|          | A | B  | C  | D  |
|----------|---|----|----|----|
| <b>1</b> | 7 | 4  | 5  | 5  |
| <b>2</b> | 8 | 14 | 11 | 9  |
| <b>3</b> | 7 | 9  | 9  | 10 |
| <b>4</b> | 7 | 3  | 5  | 6  |

**Table 2.** Experts E2’s judgments.  $X^2=14.56, \tau_a=0.034, \tau_b=0.041$

|          | A  | B  | C  | D  |
|----------|----|----|----|----|
| <b>1</b> | 4  | 6  | 0  | 1  |
| <b>2</b> | 14 | 8  | 14 | 17 |
| <b>3</b> | 7  | 10 | 10 | 10 |
| <b>4</b> | 4  | 6  | 6  | 2  |

of dependence would be acceptable only with  $X^2 > 14.68$  (never verified in the three tables). Moreover,  $\tau_a$  and  $\tau_b$  values are very close to 0.

On a total of 238 peer review answers, A obtained 33 satisfactory judgments (i.e. scored 1 or 2), B and D obtained 32, and C 30. 65/118 answers were positively scored for the physicians (A and B), and 62/120 for the systems (C and D). Although more data should probably be collected before formulating a conclusion, these not significant differences are an encouraging premise towards the statement that the RBR and the MMR systems are able to perform at an expert level. On the other hand, no significant benefit in using MMR in comparison to RBR can be demonstrated at this stage.

The number of agreements between E1 and E2 on judging the first level advisors were the following: (A) 6/29; (B) 10/30; (C) 11/30; (D) 11/30 (Mean=0.278; SD=0.07). Therefore, a lack of consensus was particularly present when evaluating colleagues; systems evaluation came out to be more uniform.

On one patient cases, both RBR and MMR were rated unsatisfactorily (with a mean score>3), with RBR considered almost unacceptable (average score=3.33) by both experts, and MMR evaluated slightly better (average score=3.25). With such patient, the correct strategy would be a complete revision of the therapy structure. The RBR system, having been built as a very conservative system [4,20], never proposes strong changes in insulin distribution and doses. On the other hand MMR, meant to overcome RBR limitations by suggesting more substantial changes, did not come out with a revision in the required direction: as a matter of fact, such patient’s cases belonged to a competence gap region, and retrieval did not allow a valuable tuning of rule parameters. In that situation the human (A and B) advice proved to be the best suited.

**Table 3.** Sum of the E1 and E2 experts’ judgments.  $X^2=5.68, \tau_a=0.006, \tau_b=0.07$

|   | A  | B  | C  | D  |
|---|----|----|----|----|
| 1 | 11 | 10 | 5  | 6  |
| 2 | 22 | 22 | 25 | 26 |
| 3 | 14 | 19 | 19 | 20 |
| 4 | 11 | 9  | 11 | 8  |

3.4 How to Improve MMR Reliability

As observed above, although MMR can be considered sufficiently safe and reliable, it does not provide a significant benefit in comparison to RBR. MMR limitation seems to be linked to the presence of competence gaps. Our guess is that MMR would largely benefit from a richer case library, a goal that would be automatically achieved by introducing the system usage into clinical practice.

Such prediction has been confirmed through a simulation study, in which we queried the therapy suggestion for a sample case, providing the MMR system with a case library whose content was progressively enlarged. We started by applying RBR (i.e. MMR when the case library is empty). Then the number of cases was increased in 5 additional steps. Each time the quality of MMR decision support was tested by applying the prescribed therapy, and by simulating 21 days of Blood Glucose data on its basis.

We verified an enhancement in the simulated patient’s metabolic condition as the case library competence augmented (see figure 2). We resorted to the M index [21] (see formula (3)), that calculates the difference on a log scale between each measured BGL ( $BGL_i$ ) and a reference value ( $BGL_0$ ).

$$M = 10^3 \sum_i (\log \frac{BGL_i}{BGL_0})^3 \tag{3}$$

4 Conclusions

Through a tight integration between CBR and RBR, we have been able to define a suitable methodology for supporting therapy planning in the context of IDDM patients management. RBR is devoted to provide information about established domain knowledge; on the other hand, CBR is able to manage and upgrade the implicit knowledge repository, during the routine clinical activities.

This methodology can be viewed in a larger KM perspective: we have been able to define an anytime continuously learning system, that exploits a naturally growing knowledge source (i.e. the case library, upgraded at each patient visit), together with formalized information. When a new case is collected, it is stored in the case library, without requiring an additional workload to physicians. At retrieval time learning takes place, as the system output is enriched

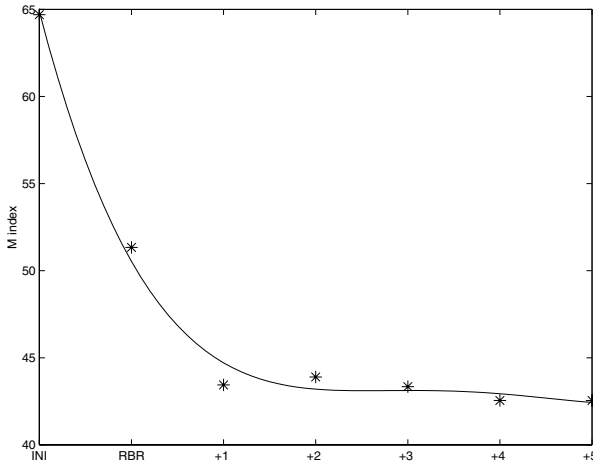


**Fig. 2.** Polynomial fitting of the M index values vs. the case library enlargement. The first M index value was calculated on BGL data simulated by applying the initial therapy. Therapy was then revised resorting to RBR (point 2), and to MMR in 5 consecutive steps (points 3 to 7), while the library competence grew. After each revision, 21 days of Blood Glucose measurements were simulated. According to the M index, the patient's metabolic condition was improved as the case library was enlarged

by the presence of additional examples. Retrieval provides information on physicians' expertise, and on their approach towards new therapeutic solutions (e.g. the use of new insulin types, such as Lyspro), thus enriching the Health Care Organization domain knowledge, without requiring a formalization of the recently acquired information within a structured knowledge base. When the case library is sufficiently competent, rule parameters are tuned on the basis of Case Based retrieval to provide decision support.

The MMR system can hence be integrated in a more general KM cycle, described in figure 3, in which the physician plays the active role of defining a new therapy, and of assessing it, relying on all the available types of information (both implicit and explicit). Therapy planning is a knowledge creation activity: the new therapy would normally be stored in the case library as a piece of implicit information. On the other hand, through this KM cycle, decision making gives birth to explicit knowledge, already analyzed in the light of domain know-how, past experience and patient's features.

This process of continuous learning, and of interoperability between the user and the system, seems to be particularly suited for the IDDM management context, in which a standard in therapy planning cannot be defined, and a lack of inter-expert consensus exists (see section 3.3).

By applying a formal evaluation methodology, we tested the classifier accuracy and the system safety and reliability, prerequisites for its introduction

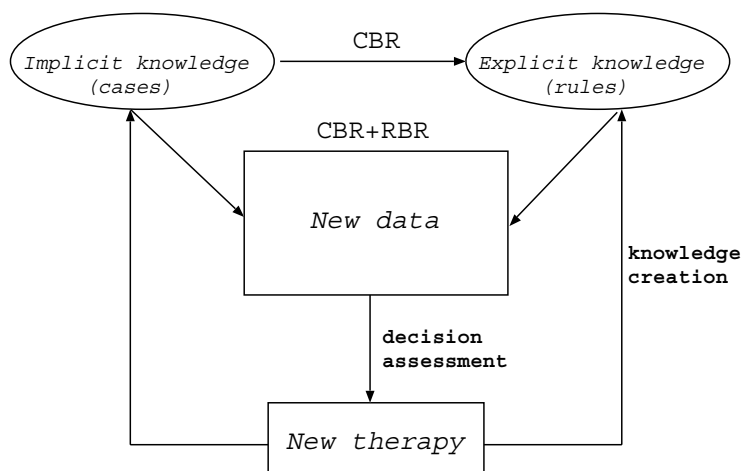


**Fig. 3.** MMR within a more general KM cycle

into clinical practice. The system came out to be reliable, and comparable to physicians' performances.

The MMR functionality, fully integrated in the T-IDDM project web-based environment, has been made available at the pediatric department of Policlinico S. Matteo Hospital in Pavia. By providing decision support capabilities within a larger KM perspective, instead of building a stand-alone tool, we believe our system could be really deployed in clinical practice, overcoming the well known weaknesses shown by the majority of decision support systems developed in medicine, due to usability, accessibility and organizational factors [22,23].

To evaluate the effectiveness of this paradigm shift, we will have to verify the effect on patients metabolic outcomes, and to address clinical impact and usability issues. The first task will be fulfilled by comparing glycated hemoglobin and daily insulin requirement before and after the introduction of the methodology. To assess usability and clinical impact, we plan to measure the intensity of the information flows among health care providers, the number of visits pre and post usage, and the time/costs gained.

## References

1. The Diabetes Control and Complication Trial Research Group. The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. *The New England Journal of Medicine* **329** (1993) 977–986. 467
2. Aha, D. W., Munoz-Avila, H. (eds.): *Proc. AAAI 99 Workshop on Exploring Synergies of Knowledge Management and CBR* (1999). 468
3. Schmidt, S., Vollrath, I. (eds.): *Proc. ICCBR 99 Workshop on Challenges for case-Based reasoning* (1999). 468

4. Bellazzi, R., Montani, S., Portinale, L., Riva, A.: Integrating Rule-Based and Case-Based Decision Making in Diabetic Patient Management. In: Proceedings of IC-CBR '99, Monastery Seon, Munich, Germany (1999). 468, 470, 472, 474
5. Montani, S., Bellazzi, R.: Integrating Case Based and Rule Based Reasoning in a Decision Support System: Evaluation with Simulated Patients. JAMIA Symposium supplement (1999) 887–891. 468, 472
6. Russell, S., Norvig, P.: Artificial Intelligence, a modern approach. Prentice Hall (1995). 468, 471
7. Bellazzi, R., Cobelli, C., Gomez, E. and Stefanelli, M.: The T-IDDM Project: Telematic management of Insulin Dependent Diabetes Mellitus. In Bracale M. and Denoth F. (eds.): Health Telematics 95 (1995) 271–276. 468
8. Bellazzi, R., Montani, S., Portinale, L.: Retrieval in a Prototype-Based Case Library: a Case Study in Diabetes Therapy Revision. In: Proceedings of EWCBR '98, Dublin, Ireland (1998). 468, 469
9. Montani, S., Bellazzi, R., Portinale, L., d'Annunzio, G., Fiocchi, S., Stefanelli, M.: Diabetic Patients Management Exploiting Case-Based Reasoning Techniques. Computer Methods and Programs in Biomedicine (to appear). 468, 469, 471
10. Kononenko, I.: Inductive and Bayesian learning in medical diagnosis. Applied Artificial Intelligence **7** (1993) 317–337. 469
11. Zelic, I., Kononenko, I., Lavrac, N., Vuga, V.: Induction of decision trees and Bayesian classification applied to diagnosis of sport injuries. In: Proceedings of IDAMAP '97 workshop, IJCAI 97, Nagoya (1997) 61–67. 469
12. Montani, S., Bellazzi, R., Portinale, L., Fiocchi, S., Stefanelli, M.: A case-based retrieval system for diabetic patients therapy. In: Proceedings of IDAMAP '98 workshop, ECAI 98, Brighton (1998) 64–70. 469
13. Wilson, D. R., Martinez, T. R.: Improved heterogeneous distance functions. Journal of Artificial Intelligence Research **6** (1997) 1–34. 469
14. Portinale, L., Torasso, P., Magro, D.: Selecting most adaptable diagnostic solutions through Pivoting-Based Retrieval. In LNAI 1266, Springer Verlag (1997) 393–402. 469
15. Larizza, C., Bellazzi, R., Riva, a.: Temporal abstractions for diabetic patients management. In LNAI 1211, Springer Verlag (1997) 319–330. 470
16. Cobelli, C., Nucci, G., Del Prato, S.: A physiological simulation model in type I Diabetes. Diabetes nutrition and Metabolism **11** (1998) 78. 472
17. Montani, S., et al.: Demonstration Phase and Demonstration of the Extensions Final Report. T-IDDM Deliverable 11.1, <http://aim.unipv.it/projects/tiddm> (1999). 472
18. Shortliffe, E. H.: Clinical decision-support systems. In: Shortliffe, E. H., Perreault, L. E., Wiederhold, G., Fagan, L. M. (eds.): Medical Informatics, Addison-Wesley (1990). 472
19. Upton, G. J. G.: The analysis of cross-tabulated data. Wiley, New York (1978). 473
20. Montani, S., et al.: Protocol-based reasoning in diabetic patient management. International Journal of Medical Informatics **53** (1999) 61–77. 474
21. Schlichtkrull, J., Munck, O., Jerslid, M.: The M-value, an index of blood glucose control in diabetics. Acta Med. Scand. **177** (1965) 95–102. 475
22. Heatefield, H. A., Wyatt, J.: Philosophies for the design and development of clinical decision-support systems. Methods of Information in Medicine **32** (1993) 1–8. 477
23. Musen, M.: Architectures for architects. Methods of Information in Medicine **32** (1993) 12–13. 477



# CBR-Based Ultra Sonic Image Interpretation

Petra Perner

Institute of Computer Vision and Applied Computer Sciences  
Arno-Nitzsche-Str. 45, 04277 Leipzig, Germany  
Email: [ibaiperner@aol.com](mailto:ibaiperner@aol.com) <http://members.aol.com/ibaiperner>

**Abstract.** The existing image interpretation systems lack robustness and accuracy. They cannot adapt to changing environmental conditions or to new objects. The application of machine learning to image interpretation is the next logical step. Our proposed approach aims at the development of dedicated machine learning techniques at all levels of image interpretation in a systematic fashion. In this paper we propose a system which uses Case-Based Reasoning (CBR) to optimize image segmentation at the low level according to changing image acquisition conditions and image quality. The intermediate-level unit extracts the case representation used by the high-level unit for further processing. At the high level, CBR is employed to dynamically adapt image interpretation.

## 1 Introduction

The existing image interpretation systems lack robustness and accuracy. They cannot adapt to changing environmental conditions or to new objects. The application of machine learning to image interpretation is the next logical step.

Our proposed approach aims at the development of dedicated machine learning techniques at all levels of image interpretation in a systematic fashion.

The proposed system [1] uses Case-Based Reasoning (CBR) to optimize image segmentation at the low level according to changing image acquisition conditions and image quality [2][3]. The intermediate-level unit extracts the case representation used by the high-level unit for further processing. At the high level, CBR is employed again to dynamically adapt image interpretation [4][5].

Case-Based Reasoning can be used where generalized knowledge is lacking but a set of cases is available. The CBR unit consists of a case base, which contains formerly processed cases. Each case has an entry in case base. The entry contains the features describing the particular case and the interpretation result. A new case is interpreted by looking up the case base for similar cases and by selecting the closest case with its interpretation result. Learning strategies enable the system to improve its system performance over time by learning the similarity or the case base according to different abstraction levels.

This strategy ensures that we can run the system without knowing the general concept description for the objects and without knowing the entire objects that should be interpreted by the system.

Our system does not rely on a propose-verify framework [6]. It also does not use CBR for the reuse of image processing knowledge [7]. We describe a novel architecture which uses CBR-learning at the low-level and high-level unit of an image interpretation system.

In section 2, we describe the proposed system architecture and the fundamental behavior of a CBR learning system. The CBR strategy of the low-level unit is described in Section 3. The feature extraction process and the high-level description of a case image are described in Section 4. In Section 5, we describe the high-level unit and give results to the system performance.

## 2 Background

Learning provides adaptation of algorithms to changes in environment and uses the experience to improve recognition performance over time. Our work aims at the development of dedicated machine learning techniques at all levels of image interpretation in a systematic fashion as shown in Figure 1. Case-Based Reasoning is used for image segmentation and the acquisition of object description. This reasoning method is used when generalized knowledge is lacking. The method works on a set of cases formerly processed and stored in a case base. A new case is interpreted by searching for similar cases in the case base. The closest case from the set of similar cases with its associated result is selected and presented to the output.

Most segmentation techniques contain numerous control parameters, which must be adjusted to obtain optimal performance. Learning optimal parameter settings requires search in a large parameter space. The parameters within most segmentation algorithm typically interact in a complex, non-linear fashion, which makes it impossible to model the parameters behavior in an algorithmic or rule-based fashion. The variation between images causes changes in segmentation results. Thus, with fixed parameter settings, the segmentation quality varies from image to image. The technique used to optimize the segmentation quality must be able to adapt to these variations.

At the high-level stage, CBR is useful to dynamically adapt image interpretation to new object and to improve performance results.

As we noted before, the main problems concerned with CBR are:

1. What is an appropriate similarity measure for the problem?
2. How can we organize a large number of cases for efficient retrieval?
3. How can we acquire and refine a new case for entry in the case base?
4. How can we generalize specific cases to a case that is applicable to a wide range of situations?

All these problems are concerned with learning. Clearly, a chosen organization for case base must be updated with every new stored case. A similarity measure that was chosen based on expert knowledge about the problem domain must not hold when more cases have been seen. In case generalization, specific cases are modified to create cases with more universal applicability and meaning.

In the next section, we will investigate how these problems can be solved for the

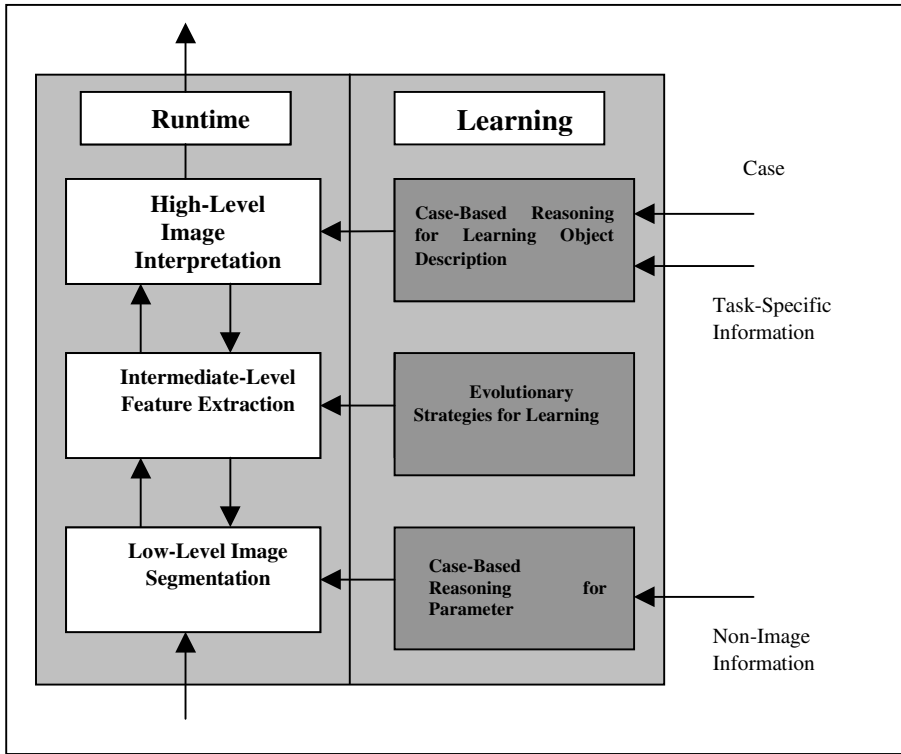


**Fig. 1 Overall Architecture of the System**

low-level and the high-level unit of an image interpretation system. For each system level we have chosen a different approach which fulfills different requirements.

### 3 Case-Based Reasoning for Low-Level Vision

Most segmentation techniques contain numerous control parameters, which must be adjusted to obtain optimum performance. The parameter selection is usually done on a large enough test data set, which should represent the entire domain well enough in order to be able to built up a general model for the segmentation. However, often it is not possible to obtain a large enough data set so that the segmentation model does not fit well to the data and needs to be adjusted to new data. Besides that, a general model does not guarantee the best segmentation for each image rather it guarantees an average best fit over the entire set of images. Another aspect goes along with changes in image quality caused by variations in environmental conditions, image devices, etc. Then the segmentation performance needs to be adapted to these changes in image quality. All that makes it necessary to equip the segmentation unit with learning

capabilities which can incrementally acquire new knowledge about the model for segmentation.

The case-based reasoning unit for parameter learning of image segmentation consists of a case base, in which formerly processed cases are stored by their original images, their non-image information (e.g. image acquisition parameters, object characteristics and so on), and their image segmentation parameters. The task is now to find the best segmentation for the current image by looking up the case base for similar cases. Similarity determination is done based on non-image information and image information.

The evaluation unit will take the case with the highest similarity score for further processing. In case there are two or more cases with the same similarity score the first appeared case will be taken. After the closest case has been chosen, the image segmentation parameter associated with the selected case will be given to the image segmentation unit and the current image will be segmented, see Fig. 2. It is assumed that images having similar image characteristics will show similar good segmentation results when the same segmentation parameters were applied to these images.

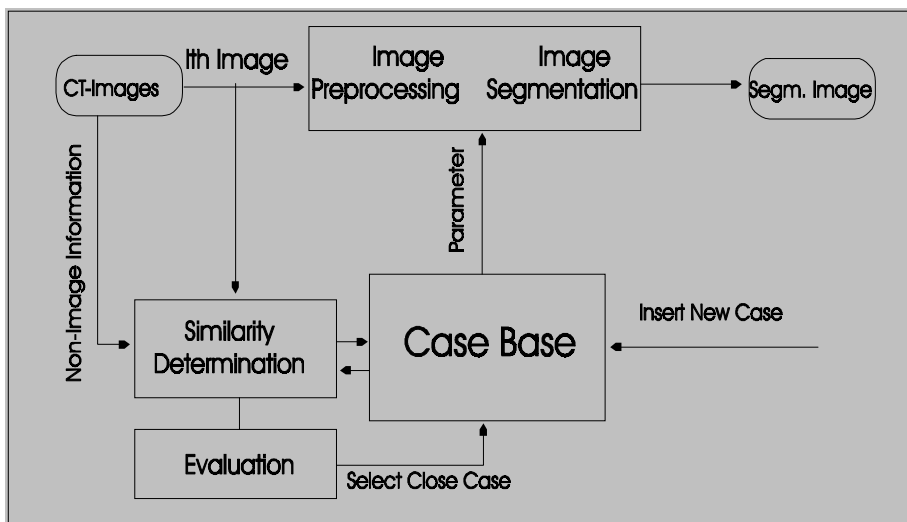


Fig. 2 Similarity-Based Image Segmentation Unit

### 3.1 Similarity Measure for Non-Image Information

We use Tversky's similarity measure [12] for the non-image information. The similarity between a Case  $C_i$  and a new case  $b$  presented to the system is:

$$S(C_i, b) = \frac{|C_i|}{\alpha|A| + \beta|D| + \gamma|M|} \quad (1)$$

$$\alpha = 1, \beta, \gamma = \frac{1}{2}$$

where  $|C_i|$  is the set of attributes in case  $C_i$ ,  $A$  is the set of corresponding attributes in case  $C_i$  and  $b$ ,  $D$  is the set of attributes having different values, and  $M$  is the set of attributes having missing values.

### 3.2 Similarity Measure for Image Information

Similarity is determined with an algorithm proposed by Zamperoni et. al. For detailed description of the algorithm we refer the interested reader to [13]. The input to the algorithm is two images that should be compared. According to the specified distance function, the proximity matrix is calculated for one pixel at position  $r,s$  in image  $A$  to the pixel at the same position in image  $B$  and to surrounding pixel within a predefined window. The same is done for the pixel at position  $r,s$  in image  $B$ . Then, clustering is performed based on that matrix in order to get the minimum distance among the compared pixel. Afterwards, the average of both values is calculated. This repeats until all pixels of both images are processed. Then from the average minimal pixel distance the distance value for the whole image is calculated and this value is given to the output. Summarizing this, the similarity is calculated in three steps: point-to-point, point-to-image, and image-to-image.

Table 1 shows the similarity based on the algorithm of Zamperoni et. al for the images shown in Figure 3.

| Image | B1_1    | B2_1    | B2_3    | B2_4    | B2_5    | B4_1    |
|-------|---------|---------|---------|---------|---------|---------|
| B1_1  | 0       | 0,19748 | 0,15646 | 0,15748 | 0,16216 | 0,32239 |
| B2_1  | 0,19748 | 0       | 0,13614 | 0,15844 | 0,16099 | 0,30203 |
| B2_3  | 0,15646 | 0,13614 | 0       | 0,10337 | 0,10571 | 0,28891 |
| B2_4  | 0,15748 | 0,15844 | 0,10337 | 0       | 0,09431 | 0,27510 |
| B2_5  | 0,16216 | 0,16099 | 0,10571 | 0,09431 | 0       | 0,28985 |
| B4_1  | 0,32239 | 0,30203 | 0,28891 | 0,27510 | 0,28985 | 0       |

Tab. 1 Similarity Values for the Images in Fig. 2

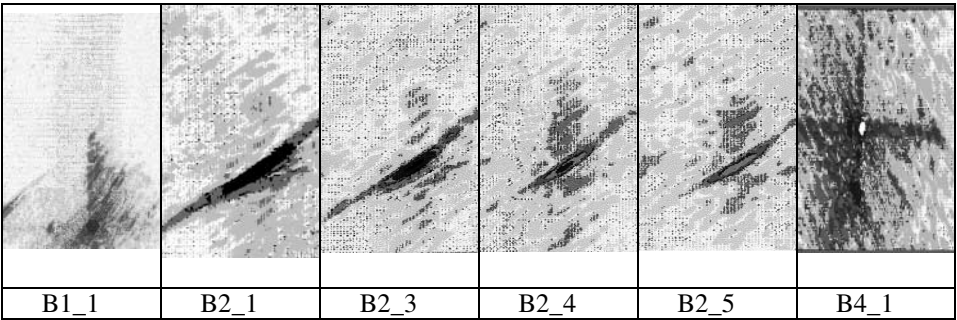


Fig. 2 Images used for Similarity Calculation shown in Tab. 1

3.3 Segmentation Parameters

Segmentation is done by thresholding in this application. The segmentation parameter, which is given to the segmentation unit, is the threshold. An example for a CBR based approach with more sophisticated segmentation using more than ten segmentation parameters is shown in [2][3] for a medical approach.

3.4 Monitoring System Performance

There are several approaches for monitoring the segmentation performance possible. The most common strategy in CBR is to let the user evaluate the result of the system. Whereas at the high-level unit such an approach may be adequate since the decision is only ‘right answer’ or ‘wrong answer’. At the low-level unit, a subjective decision might not be useful although this has widely been done in the past. To objectify the quality of the segmentation result we calculate the similarity between the gray levels of the original image and the segmented image based on Zamperoni et al dissimilarity measure [13]. The learning system will be called if the calculated value for similarity exceeds a predefined threshold. Figure 4 shows the similarity between the original image and the processed image for different thresholds. The subjective chosen threshold gives also the best similarity.

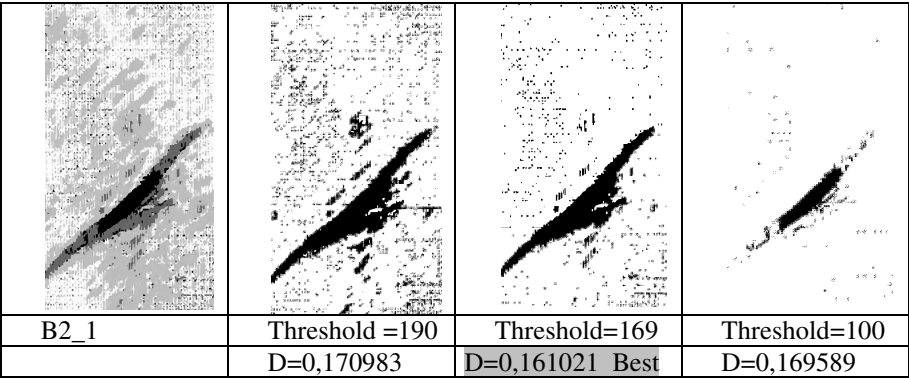


Fig. 3 Different Segmentation Results for Image B2\_1 and the Similarity Values

3.5 Learning

If the learning system was called by the evaluation unit, the run-time system will give the recent case (the image and the non-image information) to the learning system. In an off-line process, the right threshold will be chosen interactively by the user so that the similarity measure will be less than the predefined threshold. Afterwards, the segmentation parameter, the image and the non-image information are collected to a new case and stored into the case base.

In principal, all case-based management and learning functions for the case base described at the high-level unit are applicable here as well. But we restricted our work at this level to the kind of learning described above since we use a parallel system for the processing which guaranties us the efficiency in computation time.

### **3.6 Organization of Case Base and Retrieval**

The case base is organized as a flat structure. The cases are stored one behind the other in the case base. To ensure the efficiency of the retrieval process we implemented the low-level unit on a parallel system (4 node processor based on Power PC 604). The case base is partitioned into 3 equally portions. Each of these three portions is distributed to one of these three processor nodes. A query is given to each of these three nodes. Each processor calculates the similarity to each of the cases in it's case base. The similarity values are given to the fourth processor node to score out the highest similarity. The image segmentation parameters associated to the close case are given to the fourth processor. Afterwards, the original image is processed by the segmentation unit and the result is evaluated. If the segmentation result is not satisfactory, the recent case is interactively processed. The final segmentation parameter together with the original image and the non-image information are stored into one of the three processor node so that the cases are equally distributed.

## **4 Extraction of Features and High-level Description of a Case**

After the image is processed by the segmentation unit, the intermediate-level unit calculates the high-level description of the image, which is given as case representation to the CBR unit at the high-level stage. A single image, following called case, is represented by objects, their attributes (e.g. graylevel = light, grey, black; size = large, middle and small) and the spatial relations (e.g. left\_behind, right a.s.o.) between the objects. That gives a structural representation for an image represented as an image graph. Images of that kind are for example ultra sonic images for defect classification shown in Figure 3. The images were acquired by the SAFT ultra sonic system [14]. The thresholding technique is used for segmentation (s. Fig. 4), preprocessing is done by using morphological operators like dilation and erosion (s. Fig. 5) and afterwards the objects are labeled by the line coincidence method. Symbolic transformation of the numeric information of an object is done with the help of a functional model for space [15], size and grey level. The symbolic description of the image is used since it is natural to an operator to describe images in these terms.

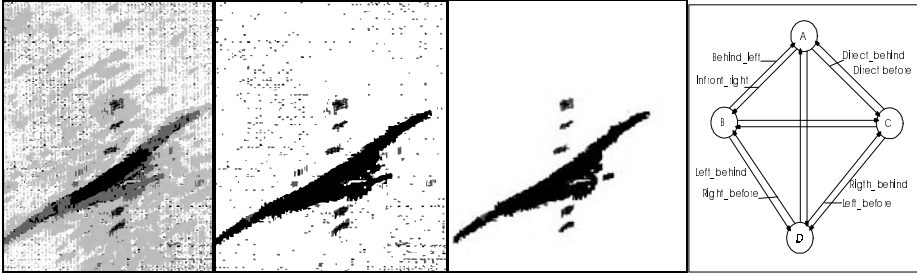


Fig. 3 Original Image

Fig. 4 Binary Image

Fig. 5 Preprocessed Image

## 5 Case Based Reasoning for Image Interpretation

The basis for the development of our system is a set of cases  $CB = \{G_1, G_2, \dots, G_p, \dots, G_n\}$ , each case is a 3-Tupel  $G_i = (N, p, q)$ , which is a structural symbolic representation of an image and a similarity measure  $\delta$  for structural representations. For the current image an image graph is extracted by image analysis. This structure is used for indexing. The interpretation of a current image  $S$  is done by case comparison: Given an image  $S = (N_s, p_s, q_s)$ , find a case  $G_m$  in the case base  $CB$  which is most similar to the current image. Output the case  $G_m$  and the stored solution (e.g. in our case, the defect name).

### 5.1 Similarity Determination

For similarity determination between our image graphs, we chose part isomorphism [4][10]. In order to handle the unsharp attributes and distortion in a graph representation we relaxed the required correspondence of attribute assignments of nodes and edges in such a way that we introduced ranges of tolerances according to the semantic terms [4].

### 5.2 Organization of Case Base

If we have many cases in case base, then it can becomes very time consuming to find the closest case to an actual case. We have seen before that we can speed up the retrieval process by using a parallel platform for our processing unit. Another way to improve the efficiency of our CBR system is to develop an index structure which allows us to find the closest case without examining all existing cases for similarity in case base. On way to impose constraints on retrieval is to index cases using a case base hierarchy [4][11]. Elements in the case base are representations between graphs. As an important relation between this graph we have considered similarity based on part isomorphism. Because of this characteristic, it is possible to organize the case base as directed graph.

In the following, we define the case base as graph that contains the before described image graphs (or only a pointer to this image graphs) in the nodes:



**Definition 5**

Given is  $H$ , the set of all image graphs.

A case base graph is a Tupel  $CB = (N, E, p)$ , with

(1)  $N \subseteq H$  set of nodes and

(2)  $E \subseteq N^2$  set of edges.

This set should show the part isomorphism in the set of nodes, meaning it should be valid

$$x \leq y \Rightarrow (x, y) \in E \text{ for all } x, y \in N.$$

(3)  $p: N \rightarrow B$  mapping of attribute assignments to the image graph (also the attribute

values for the description of the defect classes and categories).

Because of the transitivity of part isomorphism, certain edges can be directly derived from other edges and do not need to be separately stored. A relaxation of top (2) in definition 5 can be reduced storage capacity.

**5.3 Learning**

By the development and usage of the case base, we should consider the possibility that the case base may be permanently changing during the learning process. The initial case base may be built up by existing cases. Therefore a nonincremental learning procedure is required. During the use of the system new cases may be stored in the case base. They should be integrated in the already existing case base. Therefore, we need an incremental learning procedure.

Now, the task is to build up the graphs of CB in a supergraph by a learning environment. After the expert or operator has assigned the defect category or class, the system is able to find automatically an answer to the given image graph. Formally, this task is to solve permanently:

Input is:

case base Supergraph  $CB = (N, E)$  and

image graph  $x \in H$ .

Output is:

modified case base Supergraph  $CB' = (N', E')$

with  $N' \subseteq N \cup \{x\}$ ,  $E \subseteq E'$ ,

At the beginning of the learning process or the process of construction of case base  $N$  can be an empty set.

The inclusion

$$N' \subseteq N \cup \{x\}$$

says that the image graph  $x$  can be isomorphic to one in the case base contained image graph  $y$ , so  $x \leq y$  and also  $y \leq x$  hold. Then, no new node is created that means the case base is not increased.

The algorithm for the construction of the modified case base  $CB'$  can also use the circumstance that no image graph is part isomorph to another image graph if it has more nodes than the second one.

For technical help for the algorithm there are introduced a set  $N_i$ .  $N_i$  contains all image graphs of the case base CB with exactly  $i$  nodes. The maximal number of nodes of the image graph contained in the case base is  $k$  then it is valid:

$$N = \bigcup_{i=1}^k N_i$$

The image graph, which has to be included in the case base, has  $l$  nodes ( $l > 0$ ). By comparing the current image graph with all in the case base contained graphs, we can make use of transitivity of part isomorphism for the reduction of the nodes that has to be compared.

### Algorithm

```

E' := E;
Z := N;
for all  $y \in N_l$ 
if  $x \leq y$  then [ CB' := CB; return];
N' := N  $\cup$  {x};
for all  $i$  with  $0 < i < l$ ;
    for all  $y \in N_i \setminus Z$ ;
        for all  $y \leq x$  then [ Z := Z  $\setminus$  {u | u  $\leq$  y, u  $\in$  Z};
                                E' := E'  $\cup$  { (y,x) }];

for all  $i$  with  $l < i \leq k$ 
    for all  $y \in N_i \setminus Z$ 
        if  $x \leq y$  then [ Z := Z  $\setminus$  {u | u  $\leq$  y, u  $\in$  Z };
                                E' := E'  $\cup$  { (x,y) }];

```

If we use the concept for uncertainty handling, then we can use the algorithm without any changes. But we should notice that for each group of image graphs that is approximately isomorphic, the first occurred image graph is stored in the case base. Accidentally, this can be a "bad" image graph. Therefore, it is better to calculate of every instance and each new instance of a group a prototype and store this one in the case base. Thus, we can learn the appropriate instance\_of. The approach described allows us to learn new cases and store the new case into the case base according to the similarity relation to the already existing cases in case base. If we relax the part isomorphism relation to the concept for uncertainty handling then in each node of the hierarchy is stored a class of cases, which is represented by a prototype. The generalization ability of the algorithm can be described as follows: Given a set of structural relations and two cases  $x$  and  $y$ , then construct a case  $g$  such that  $g \succ x$  and  $g \succ y$ . Then  $g$  is the most specific common substructure of the two cases, which gives a generalization of the two cases.

In this learning approach, the learned hierarchy depends on the chosen threshold for similarity. An approach where no threshold is necessary is shown in our further work in [16].

## 5.4 Indexing and Retrieval

Retrieval is done as follows: The current case is matched according to the case base hierarchy. The case class with the maximal structural similarity is indexed. Among the members of the indexed case class the closest cases is determined.

## 6 Evaluation of the System

Accuracy (  $A = (\text{number of right recall} / \text{number of samples}) * 100$  ) was calculated based on 100 samples [8]. The information „no similar case“ was only taken as an error if there was a similar case in case base but the system could not call it based on the chosen similarity. Among this data set were 5 cases which are not related to any case in case base. Accuracy was 95%.

## 7 Conclusion

We have shown how Case-Based Reasoning can be employed for the low-level and high-level unit of an image interpretation system. It can be used for reasoning as well as for learning. At the low-level stage we can learn incrementally a better system performance of the segmentation unit and at the high-level stage a more compact concept for the case descriptions.

## References

- [1] P. Perner, Ultra Sonic Image Segmentation and Interpretation based on CBR, In: M.H. Göker (ed.), 8<sup>th</sup> German Workshop on Case-Based Reasoning GWCBR2000, p.133-145.
- [2] P. Perner, An Architecture for a CBR Image Segmentation System, Journal on Engineering Application in Artificial Intelligence, Engineering Applications of Artificial Intelligence, vol. 12 (6), 1999, p. 749-759
- [3] P. Perner, An Architecture for a CBR Image Segmentation System, In: K.-D. Althoff, R. Bergmann, and K. Branting (Eds.) Case-Based Reasoning Research and Development, Inai 1650, Springer Verlag, 1999, pp. 525-535
- [4] Perner P., Paetzold W. An Incremental Learning System for Interpretation of Images. In: D. Dori and A. Bruckstein (eds.). Shape, Structure, and Pattern Recognition. World Scientific Publishing Co., 1995, pp 311-323.
- [5] Perner P., Paetzold W. An Incremental Learning System for Image Interpretation. HTWK Report 5/93.

- [6] M. Grimnes and A. Amond, A Two Layer Case-Based Reasoning Architecture for Medical Image Understanding, In: I. Smith and B. Faltings (Eds.), *Advances in Case-Based Reasoning*, Springer Verlag 1996, Lnai 1168, pp. 164-178.
- [7] V. Ficet-Cauchard, C. Porquet, and M. Revenu, CBR for the Reuse of Image Processing Knowledge: A Recursive Retrieval/Adaption Strategy, In: K.-D. Althoff, R. Bergmann, and L.K. Branting (Eds.), *Case-Based Reasoning Research and Development*, Springer Verlag 1999, Lnai 1650, pp.438-452.
- [8] Perner P., Anderson P., Summner D., Kyle J. An Application of Case-based Reasoning in Test Process Diagnosis. In *Proceedings of the IBM ITL Conference on Expert Systems*, Yorktown Heights USA, 1992, pp 73-85, Plenary Talk.
- [9] Perner P. Ultra Sonic Image Interpretation for Non-Destructive Testing. In *Proceedings of the IAPR Workshop on Machine Vision Applications*, 1996, pp 552-554.
- [10] Schlesinger M.I. *Mathematical Tools of Picture Processing*. Naukowa Duma, Kiew 1989.
- [11] Perner P. Case-Based Reasoning for Image Interpretation. In: V. Hlava\_ and R. Šárka (eds.). *Computer Analysis of Images and Patterns*. Springer Verlag, 1995, pp 532-537.
- [12] Tversky A. Feature of Similarity. *Psychological Review* 1977; 84(4): 111-147.
- [13] Zamperoni and V. Starovoitov, „How dissimilar are two gray-scale images“, In *Proc. of 17. DAGM Symposium 1995*, Springer Verlag, pp.448-455
- [14] Schmitz V. Ultra Sonic Imaging Techniques. In *Proceedings of the DGzfp- Annual Conference*, 1989.
- [15] Hernandey D. Relative Representation of Spatial Knowledge: The 2-D Case. Report FKI 135-90, Aug. 1990, TU München
- [16] P. Perner, Different Learning Strategies in a Case-Based Reasoning System for Image Interpretation, *Advances in Case-Based Reasoning*, B. Smith and P. Cunningham (Eds.), LNAI 1488, Springer Verlag 1998, S. 251-261.

# Evaluation of Strategies for Generalised Cases within a Case-Based Reasoning Antibiotics Therapy Advice System

Rainer Schmidt and Lothar Gierl

Institut für Medizinische Informatik und Biometrie, Universität Rostock  
Rembrandtstr. 16 / 17, D-18055 Rostock, Germany  
{rainer.schmidt,lothar.gierl}@medizin.uni-rostock.de

**Abstract.** For an intensive care unit, we have developed an antibiotics therapy advice system, called ICONS. To speed-up the process of finding suitable therapy recommendations we have applied Case-based Reasoning techniques. As in a medical expert system all required information should always be up-to-date, a static case base is inappropriate. New cases should be incrementally incorporated into the case base and outdated ones should be updated or erased. For reasons of space limitations and of retrieval time an indefinite growth of the case base should be avoided. To fulfil these requirements we propose to generalise from single cases to prototypical cases and to erase redundant cases. In this paper, we mainly focus on results of extended experiments with generation strategies of generalised cases (prototypes). Additionally, we compare measured retrieval times for two indexing retrieval algorithms: For simple indexing and Tree-Hash retrieval.

## 1 Introduction

For an intensive care unit (ICU), we have developed an antibiotics therapy advice system. It is called ICONS. To speed-up the process of finding suitable therapy recommendations we have applied Case-based Reasoning (CBR) techniques.

In contrast to some other application domains, medicine is characterised by some specific demands - especially in an ICU setting. A medical knowledge-based system should respond extremely fast. All required information should always be up-to-date. So, the idea of constructing an optimal static case base [1] is not appropriate in the medical domain, but new cases should be incrementally incorporated into the case base and outdated ones should be updated or erased. However, for reasons of space limitations and of retrieval time an indefinite growth of the case base should be avoided. In MacRad [2], a radiology image retrieval system, this happens by collecting new cases and incorporating some of them after periodical expert consultations. The expert has to decide which new cases should replace which old ones. So, the case base is of a fixed size and the expert should attempt to find an optimal set of cases. However, the consultation effort is high and disadvantages of rule-based systems, which have to be updated manually, may here occur again (e.g.

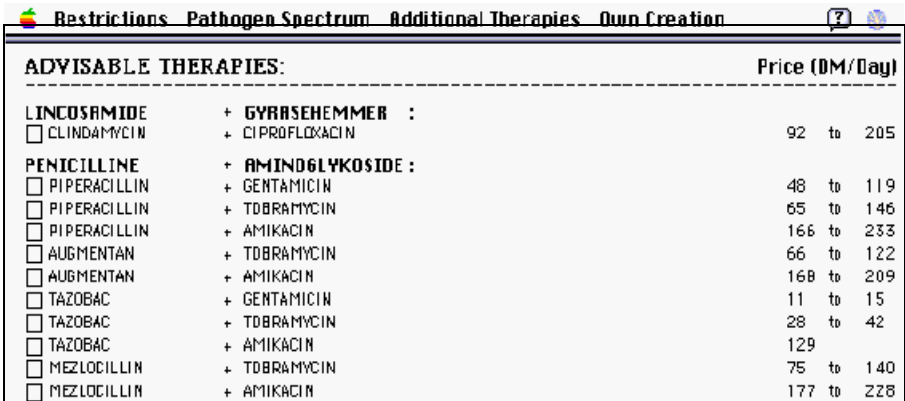
the expert may leave). Furthermore, between the expert consultations the case base does not reflect current developments. We propose a different idea, namely to generalise from single cases to prototypes and to erase redundant cases. Abstracted cases have some advantages: The case base can easily be structured, a retrieval speed-up may occur, and new knowledge may be discovered [3]. Furthermore, the use of prototypical cases is popular in medical applications (e.g. [4, 5]), because the decision making of physicians circles around typical and exceptional cases [3].

In this paper, we mainly focus on results of extended experiments with prototype generation strategies (and different settings of threshold parameters within them). Additionally, we present measured retrieval times for two indexing retrieval algorithms. We compare a simple indexing algorithm with the Tree-Hash retrieval algorithm developed by Stottler, Henke and King [6], which was originally used in ICONS.

## 2 The Application: Antibiotics Therapy Advice

Severe bacterial infections still are a life threatening complication in intensive care medicine correlated with a high mortality [7]. Identification of bacterial pathogens often is difficult. It normally requires at least 24 hours to identify the pathogen that is responsible for an infection and at least another 24 hours to find out which antibiotics have therapeutic effects against the identified pathogen. To not endanger the patient, physicians often have to start an antimicrobial therapy before the responsible pathogen and its sensitivities are determined. This sort of antibiotic therapy is called "calculated" in contrast to a "selective" therapy, which is used when microbiological results are already available.

The main task of our adviser is to present suitable calculated antibiotics therapy advice (Fig. 1.) for intensive care patients, who have developed a bacterial infection as an additional complication.



| ADVISABLE THERAPIES:                  |                    | Price (DM/Day) |
|---------------------------------------|--------------------|----------------|
| <input type="checkbox"/> LINCOSAMIDE  | + GYRASEHEMMER :   |                |
| <input type="checkbox"/> CLINDAMYCIN  | + CIPROFLOXACIN    | 92 to 205      |
| <input type="checkbox"/> PENICILLINE  | + AMINOGLYCOSIDE : |                |
| <input type="checkbox"/> PIPERACILLIN | + GENTAMICIN       | 48 to 119      |
| <input type="checkbox"/> PIPERACILLIN | + TOBRAMYCIN       | 65 to 146      |
| <input type="checkbox"/> PIPERACILLIN | + AMIKACIN         | 166 to 233     |
| <input type="checkbox"/> AUGMENTAN    | + TOBRAMYCIN       | 66 to 122      |
| <input type="checkbox"/> AUGMENTAN    | + AMIKACIN         | 168 to 209     |
| <input type="checkbox"/> TAZOBAC      | + GENTAMICIN       | 11 to 15       |
| <input type="checkbox"/> TAZOBAC      | + TOBRAMYCIN       | 28 to 42       |
| <input type="checkbox"/> TAZOBAC      | + AMIKACIN         | 129            |
| <input type="checkbox"/> MEZLOCILLIN  | + TOBRAMYCIN       | 75 to 140      |
| <input type="checkbox"/> MEZLOCILLIN  | + AMIKACIN         | 177 to 228     |

**Fig. 1.** Advisable antibiotics therapy presentation. In the left column a list of advisable therapeutics is presented. The user may choose one of them. The right column shows the prices per day for the therapies. By clicking the buttons in the top line the user can get additional information, e.g. about the covered pathogen spectrum, contraindication per antibiotic etc.

As physicians cannot wait for the laboratory results of such critical patients, we use an expected pathogen spectrum based on medical background knowledge. This spectrum should be completely covered by each advisable antibiotics therapy. Furthermore, as advice is needed very quickly, we speed-up the process of computing advisable antibiotics therapies by using Case-based Reasoning methods. We search for a similar previous patient and transfer the suggested therapies made for his situation to the current patient. These previous suggestions are adapted to be applicable to the new medical situation of the current patient.

## 2.1 Strategy for Selecting Advisable Antibiotic Therapies

As ICONS is not a diagnostic system, we do not attempt to deduce evidence for diagnoses based on symptoms, frequencies or probabilities, but instead pursue a strategy that can be characterised as follows: Find all possible solutions (therapy recommendations) and reduce them by using the patient's contraindications and the complete coverage of the calculated pathogen spectrum (establish-refine strategy).

First, we distinguish among different groups of patients (infection acquired in- or outside the ward resp. the hospital, immunocompromised patients). A first list of antibiotics is generated by a susceptibility relation, which for each group of pathogens provides all antibiotics that usually have therapeutic effects. This list contains all those antibiotics that can control at least a part of the potential pathogen spectrum. We obtain a second list of antibiotics by reducing the first one by applying two constraints: The patient's contraindications and the desired sphere of activity. Using the antibiotics of this second list, we try to find antibiotics that under consideration of the expected susceptibility cover the whole pathogen spectrum individually.

Except for some community-acquired infections, monotherapies have to be combined with synergistic or additive effecting antibiotics. If no adequate single therapy can be found, we use combination rules to generate combinations of antibiotics. Each of them must be tested for the ability to cover the whole expected spectrum.

## 3 Case-Based Reasoning in ICONS

In this application the main argument for using CBR methods is to speed-up the process of finding adequate therapies. We shorten the above described strategy of selecting advisable antibiotic therapies by searching for a similar case, retrieving its suggested therapies and by adapting them concerning the contraindications of the current patient.

### 3.1 Retrieval

The retrieval consists of three steps. First we select that part of the case base, in which all cases share two attributes with the current patient: The group of patients and

the infected organ system. This means a selection of the appropriate prototype tree (see chapter 4.). Subsequently, we apply the Hash-Tree-Retrieval-Algorithm of Stottler, Henke, and King [6] for nominal valued contraindications. Furthermore, we use an adaptability criterion, because not every case is adaptable [8]. As the attributes used for the retrieval are the contraindications, which work as constraints for the set of possible antibiotics suggestions, it is obvious, that a former case that has contraindications the current patient does not share should not be used. To guarantee this condition the adaptability criterion has to be checked during the retrieval.

### 3.2 Comparison of Two Indexing Retrieval Algorithms

The attributes are contraindications (e.g. a penicillin-allergy or a pregnancy, Fig.2.) As these are unordered nominal values, we did not consider some retrieval algorithms like CBR retrieval nets [9], which are appropriate for ordinal nominal values, or k-d trees [10], which are appropriate for metric values, but only indexing algorithms.

|                                                            |                                           |                                           |                                        |
|------------------------------------------------------------|-------------------------------------------|-------------------------------------------|----------------------------------------|
| <b>Allergy to:</b>                                         | <input type="checkbox"/> Penicillin       | <input type="checkbox"/> Aminoglykosid    | <input type="checkbox"/> Cephalosporin |
|                                                            | <input type="checkbox"/> Tacobactam       | <input type="checkbox"/> Clavulan acid    | <input type="checkbox"/> Imipenem      |
| <input type="checkbox"/> <b>Pregnancy / Nursing Period</b> |                                           |                                           |                                        |
| <input type="checkbox"/> <b>CNS Disease</b>                |                                           |                                           |                                        |
| <input type="checkbox"/> <b>Inner Ear Damage</b>           |                                           |                                           |                                        |
| <input type="checkbox"/> <b>Liver Malfunction</b>          |                                           |                                           |                                        |
| <b>Kidney Malfunction:</b>                                 |                                           |                                           |                                        |
|                                                            | <input type="checkbox"/> Haemo-Filtration | <input type="checkbox"/> Haemo-Dialysis   |                                        |
| <b>Blood:</b>                                              | <input type="checkbox"/> Leucopenie       | <input type="checkbox"/> Thrombocytopenie |                                        |

Fig. 2. Input of possible contraindications of a patient

Originally we applied the Tree-Hash retrieval algorithm developed by Stottler et al. [6], which is advantageous for large case bases. Recently we have compared it with a simple indexing algorithm. The results show that the answer to the question: Which algorithm works faster? depends on the definition of large.

**Tree-Hash retrieval.** The Tree-Hash retrieval algorithm was designed for retrieval of cases whose attributes have qualitative values. In a pre-processing step a tree is constructed. For each possible combination of the dimensions a node is set up. For example, in the two- dimensional case with dimensions d1 and d2, the possible dimension combinations are {d1, d2, d1d2}. The algorithm attempts retrieval based on each of these combinations and computes a similarity measure. In this example, a first retrieve is performed based on d1, meaning that d1 is the only dimension that



must produce an exact match. The same is done for d2. Retrieval based on d1d2 indicates that both dimensions must match the dimensions of a case in the case base.

The algorithm is very fast, when the number of dimensions is low (up to about fifteen). Retrieval time does not increase with the number of cases if a hashing scheme is used.

The Tree-Hash pre-processing step generates all possible combinations of attributes and represents them as nodes in a tree. The dimensions are placed in decreasing order of importance. The following algorithm generates the tree structure displayed in Fig.3, when the number of dimensions,  $D$ , equals three.

```

Root = d1
call tree (Root, 1)

function tree (t, n)
  if n = D then return t
  else
    leftson (t) = call tree (dn+1 concatenated to back of t, n+1)
    rightson (t) = call tree (dn+1 concatenated to back of t without its last
    dimension, n+1)

```

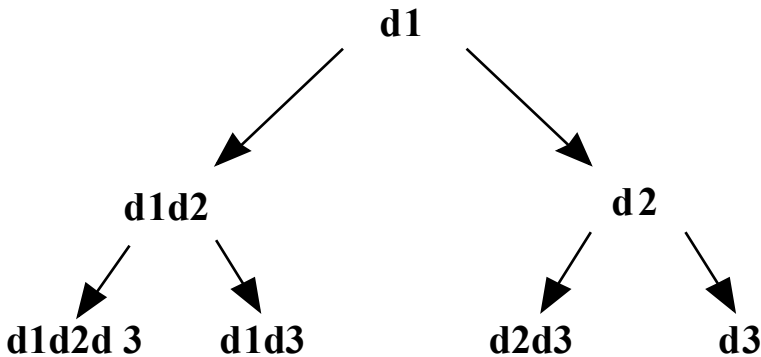


**Fig. 3.** Example Hash-Tree for three dimensions

To provide efficient lookup, a unique pointer is created for each case and placed in a hash table exp.  $(2,D) - 1$  times, once for each possible combination of attributes, ignoring order. When a node is visited, the case is hashed using the dimensions represented by the node. The result of this process is a large hash table, which encompasses all the cases, organised by every possible ordering of dimensions. Retrieval can now simply consist of visiting the nodes of the tree and performing a hash look-up based on the dimensions of the node.

Instead of visiting every node in the tree, it is more efficient to search the tree in a specified order to eliminate many of the hash retrievals. Traversal of the tree begins at the root. At each node, an attempt is made to hash retrieve a case based on the dimensions of that node. If the retrieval is successful, it goes on down the left branch.

If unsuccessful, it goes on down the right branch, because in the left branch no success is possible. This process is continued until a leaf node is reached. At this point, the computed similarity of the node of the last successful retrieval is stored. Then follows a backtrack to the last successful node and it goes down its right son. This process is repeated until there are no successful nodes left to backtrack to. The stored similarity values are compared and the best one is selected. This process can be shortened. During the pre-processing, the highest possible score of each node is stored. As the highest possible score at any node is the maximum of the possible scores of its subnodes, nodes and their subnodes which have a lower highest possible score than the already reached similarity value do not have to be visited.

**Comparison with a simple indexing algorithm.** Simple indexing does not construct a Hash-Tree, but hashes cases directly. We used two tables. In the first one we store the attribute values with pointers to lists of cases with corresponding values. The second table is set up, when similar cases are searched, it contains information like "case-no. -> sum of (so far) indexed weights". The cases with the same values as the query case are looked-up in the first table and their weights are increased in the second one. At the end, the cases with highest weights are retrieved. For both algorithms we used a test set of 100 cases, which were incrementally incorporated into the system (table 1).

**Table 1.** Retrieval times for Hash-Tree and simple indexing retrieval algorithms

| Cases (No.) | Average Retrieval time<br>for Hash-Tree | Average Retrieval Time<br>for Simple Indexing |
|-------------|-----------------------------------------|-----------------------------------------------|
| 2-15        | 0.233                                   | 0.043                                         |
| 16-30       | 0.231                                   | 0.044                                         |
| 31-45       | 0.234                                   | 0.045                                         |
| 46-60       | 0.233                                   | 0.046                                         |
| 61-75       | 0.232                                   | 0.047                                         |
| 76-80       | 0.233                                   | 0.047                                         |
| 81-85       | 0.231                                   | 0.048                                         |
| 86-90       | 0.233                                   | 0.048                                         |
| 91-95       | 0.231                                   | 0.049                                         |
| 96-100      | 0.232                                   | 0.049                                         |

The results can be summarised as follows: Simple indexing works much faster than the Hash-Tree algorithm. However, for simple indexing the retrieval time increases with the number of stored cases, approximately by 0.001 seconds per 15 cases. As Hash-Tree does not depend on the number of stored cases, both algorithms should have equal retrieval times when about 3.000 cases are stored. However, this result depends on our concrete implementation in Macintosh Common Lisp (MCL 3.0). As Lisp does not support any pointer concepts, which are useful for operations on trees, graphs etc., an implementation in other programming languages like C or PASCAL might come up with better results for the Hash-Tree algorithm.

So, the overhead for Hash-Tree operations has its positive effect for huge case bases. In comparison to simple indexing "huge" begins in our implemented application version with few (approximately about three) thousands of stored cases.

### 3.3 Adaptations

In the antibiotics therapy adviser three different sorts of adaptations occur: CBR adaptation to obtain sets of calculated advisable therapies for current patients (Fig. 1. shows a presentation of such a set), adaptation of chosen therapies to laboratory findings and a periodical update of laboratory information used by the system. The latter two sorts do not really belong to the CBR paradigm, but are rather typical for medical applications.

During the retrieval we use a criterion that guarantees that the retrieved case does not have any additional contraindications in comparison to the current case. So, the adaptation of a similar case is just a transfer of the set of advisable therapies and if necessary a subsequent reduction of this set by additional contraindications of the current case.

## 4 Prototypes

As in an incrementally working system the number of cases increases continuously, storing each case would slow down the retrieval time and exceed any space limitations. So we decided to structure the case base by prototypes and to store only those cases that differ from their prototype significantly. Though the general use of prototypes was early introduced in the CBR field [e.g. 11, 12], it is still mainly applied in the medical domain [e.g. 4, 5], because the reasoning of physicians circles around (proto-) typical and rather exceptional cases.

The prototype idea is founded on empirical research [13], which indicates that people consider cases to be more "typical" when the number of features between the presented case and the "normal" case increases. In diagnostic applications prototypes correspond to typical diseases or diagnoses. So, for antibiotic therapies prototypes are expected to correspond to typical antibiotic treatments associated with typical clinical features of patients. However, as the attributes are contraindications that are not responsible for the generation, but the restriction of the solution set, this is only partly true.

**Selection of a prototype tree.** We do not have just one prototype tree, but a wood of trees, which are independent from each other. An own tree can be generated for each affected organ system combined with each group of patients. That means nearly 100 possible prototype trees for about 20 organ systems and 5 patient groups. We generate them only if required dynamically. For example a tree for "community acquired kidney infections" will be generated as soon as the first data input of a patient occurs who has a kidney infection acquired outside a hospital.

So, all cases within the same prototype tree belong to the same group of patients, the same organ system is affected and therefore the same expected pathogen spectrum deduced from background knowledge has to be covered. The cases within a prototype tree are only discriminated from each other by their different contraindications. These are antibiotic allergies, reduced organ functions (e.g. kidney and liver), specific diagnoses (e.g. acoustic distortion or diseases of the central nervous system), special blood diseases, pregnancy and the patient's age (Fig.2.).

## 5 Experiments with Prototype Generation Strategies

The general idea of our concept of prototypes is to keep them always up-to-date and that they should contain the (proto-) typical features of their cases. We have tested two contrasting policies for erasing "redundant" cases and a strategy of keeping all cases. The aim of our experiments was firstly to find a strategy that fits best for two contrasting purposes, to find many adaptable cases or prototypes and to require few work space, and secondly to find good settings for the threshold parameters.

The usual idea is that cases without additional features in comparison to their prototype are redundant, because they do not contain any extra information. However, in our application the attributes are contraindications, which are not used to generate a solution, but to restrict a solution set. This means they are applied as constraints. A case with less contraindications than its prototype has a greater chance to be adaptable to a query case, because only a case without additional contraindications in comparison to the query case is adaptable.

So, we have tested both opposing strategies: Firstly, erasing cases without additional attributes, and secondly, erasing only cases with additional attributes. Additionally, we have tested a strategy without erasing any cases at all.

Without any prototypes for 51 of the 75 cases a similar adaptable case can be found. As prototypes can be treated as cases this number can be overstepped. The workspace amount without any stored case is about 2.248 MB for all three strategies.

The argument about the workspace might seem to be unreasonable, because the differences are just about 40 KB. However, we performed our tests in just one of more than 100 parallel areas. 75 cases in each area might lead to differences between the strategies of up to 4 MB. So, this means the question whether our system should require about 12 or about 16 MB workspace (the alternative, keeping the cases outside the system and incorporating them only when necessary, would extend the retrieval time). Of course, problems should not occur until the number of cases per area exceeds 75.

**The overall prototype generation strategy.** At first, all cases are stored below the prototypes they belong to. If after storing a new case below a prototype the threshold "number of cases" is reached, the prototype will be "filled". This means, that every contraindication which occurs in the cases belonging to this prototype at least as often as the second threshold "minimum frequency" will be included into the prototype. Subsequently, the "filled" prototype can be treated like a case. The same as for cases holds for prototypes: Each contraindication restricts the set of advisable therapies. The contraindications of a prototype are those that occur most often within its cases. From the viewpoint of frequency they are the typical ones.

When later on a new case is added to an already filled prototype, its frequency table, which contains information about the frequency of the contraindications of its cases, has to be updated and if necessary the contraindications of the prototype have to be recomputed. If the (re-) computed contraindications of the prototype change, the suggested antibiotic therapies have to be recomputed too. All cases must be inspected again for their need to be stored.

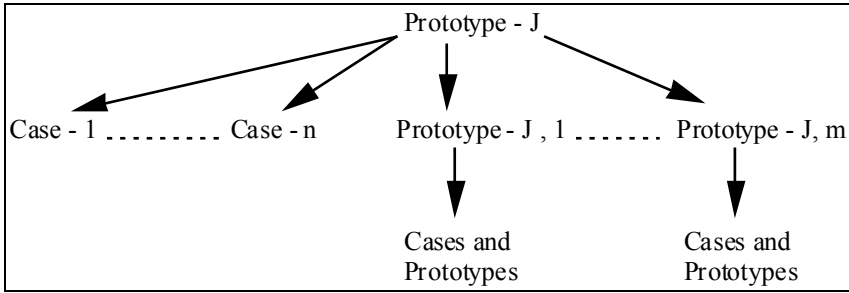


**Fig. 4.** Possible relationships of a prototype

Below an already existing prototype we create an "alternative" prototype if enough cases exist for the latter, which have at least one contraindication in common the already existing prototype does not include. We construct this new alternative prototype from those cases that share at least one from the already existing prototype deviating contraindication. We place this new prototype in the hierarchy directly below the existing prototype (a part of a possible prototype hierarchy is shown in Fig. 4.). New "alternative" prototypes differ from their superior prototypes by their contraindications and therefore also by their sets of advisable antibiotic therapies.

## 5.1 Strategy A: Erasing Cases without Additional Attributes

We have tested the strategies with 75 cases, which were incrementally incorporated into the system. For strategy A, we varied the threshold parameter "number of cases", which indicates how many cases are necessary to generate a prototype. The second threshold parameter "relative frequency" was set to 33 %, which means that an attribute value is incorporated into a prototype if at least a third of its cases have this value.

The results (table 2) can be summarised as follows: The more cases are necessary to generate a prototype (increase of "number of cases") the higher the number of stored cases and the higher the number of retrieved adaptable cases. After a while there is only little to be gained by increasing this threshold parameter any further (4th setting). Surprising is the big increase of the number of retrieved adaptable cases in the second setting compared with the first one. This cannot be simply explained by only four additionally stored cases, but by two phenomena: Those cases are erased that have no additional information (contraindications) in comparison to their prototype. This means they have better chances to be adaptable to future query cases. Secondly, in the other settings the fewer prototypes are generated later and consequently cases are erased later as well.

One idea of using prototypes is the hope to reduce the storage and workspace amount. In the tests for strategy A this does not really happen, because the storage amount of prototypes is bigger than of cases. Prototypes contain some additional information: The intersection of advisable therapies of its cases (cases only contain additional specific therapy suggestions), observed frequencies of contraindications of their cases etc.

**Table 2.** Test results for strategy A

|                                        | 1. Setting<br>number of<br>cases = 2 | 2. Setting<br>number of<br>cases = 3 | 3. Setting<br>number of<br>cases = 4 | 4. Setting<br>number of<br>cases = 5 |
|----------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|
| Workspace Amount:<br>after 75 cases    | in Mbytes<br>2.392                   | in Mbytes<br>2.390                   | in Mbytes<br>2.401                   | in Mbytes<br>2.402                   |
| Number of Prototypes                   | 9                                    | 7                                    | 8                                    | 8                                    |
| Number of stored Cases                 | 53                                   | 57                                   | 62                                   | 63                                   |
| Number of erased<br>Cases              | 22                                   | 18                                   | 13                                   | 12                                   |
| Number of retrieved<br>adaptable Cases | 12                                   | 26                                   | 31                                   | 31                                   |

**5.2 Strategy B: Erasing Cases with Additional Attributes**

Our idea for this strategy was to keep those cases in the case base that have a higher chance to be adaptable. That means cases with few contraindications. So, we attempted a strategy opposite to strategy A, namely erasing cases with additional information (contraindications) to their prototype. As many cases are erased, we set the threshold parameter "number of cases" to the value two. Here, we varied the second parameter "relative frequency", which determines the frequency contraindications have to be observed among the cases to be incorporated into a prototype.

The difference between the results for both settings is rather small. With a smaller relative frequency (2nd setting) more contraindications are incorporated into the prototypes. So, fewer cases have additional contraindications and consequently fewer cases are erased and fewer prototypes are generated. The workspace amount is nearly the same and the number of retrieved adaptable cases is exactly the same for both settings.

**Table 3.** Test results for strategy B

|                                        | 1. Parameter setting:<br>relative frequency =<br>33% | 2. Parameter setting<br>relative frequency =<br>25% |
|----------------------------------------|------------------------------------------------------|-----------------------------------------------------|
| Workspace Amount:<br>after 75 cases    | in Mbytes<br>2.373                                   | in Mbytes<br>2.368                                  |
| Number of generated<br>Prototypes      | 9                                                    | 6                                                   |
| Number of stored Cases                 | 20                                                   | 25                                                  |
| Number of erased Cases                 | 55                                                   | 50                                                  |
| Number of retrieved<br>adaptable Cases | 14                                                   | 14                                                  |

In comparison to strategy A, it is noticeable that about the same number of prototypes have been generated, but much more cases have been erased. Though those cases remain in the case base that have a bigger chance to be adaptable, the number of retrieved adaptable cases slightly increases in comparison to the first setting of strategy A, but is not nearly as good as in the other settings of strategy A.

So, the idea of keeping those cases that are easily adaptable results in such a small case base that only few adaptable cases can be retrieved.

### 5.3 Strategy C: All Cases Remain in the Case Base

For strategy C we have tested the same settings as for strategy A. It can be seen that by keeping all cases a lot more adaptable cases can be retrieved comparing the settings of strategy C with the corresponding settings of strategy A, while the workspace amount increases only slightly. As two cases are sufficient in the first setting, many prototypes are generated, the workspace amount increases correspondingly. A bit surprising seems the fact that less adaptable cases are retrieved, but as a hierarchy with three levels of prototypes has been generated and the prototypes are treated as cases, on each level the right prototype has to be determined to be the most similar case.

Really surprising is the big increase of retrieved adaptable cases in the third setting. There are two possible explanations. Firstly, as the number of generated prototypes decreases the prototype hierarchy is simpler and it is easier to find the appropriate case. Secondly, and probably the main reason, as the number of cases necessary to generate a prototype is higher (= 4), more cases are considered when a generated prototype is filled and consequently fewer contraindications are incorporated into the prototype. This means the prototypes themselves become more adaptable. However, when the number of generated prototype decreases, they are missed to be used for adaptation (4th setting).

**Table 4.** Test results for strategy C

|                                        | 1. Setting<br>number of<br>cases = 2 | 2. Setting<br>number of<br>cases = 3 | 3. Setting<br>number of<br>cases = 4 | 4. Setting<br>number of<br>cases = 4 |
|----------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|
| Workspace Amount:<br>after 75 cases    | in Mbytes<br>2.439                   | in Mbytes<br>2.426                   | in Mbytes<br>2.421                   | in Mbytes<br>2.419                   |
| Number of Prototypes                   | 19                                   | 10                                   | 8                                    | 7                                    |
| Number of stored Cases                 | 75                                   | 75                                   | 75                                   | 75                                   |
| Number of erased Cases                 | 0                                    | 0                                    | 0                                    | 0                                    |
| Number of retrieved<br>adaptable Cases | 29                                   | 32                                   | 52                                   | 51                                   |

### 5.4 Summary of the Test Results

Keeping all cases in the case base increases the storage and workspace amount, but increases the number of retrieved adaptable cases dramatically. Too many prototypes

should be avoided, because a complex hierarchy structure results in difficulties to find the desired case. This means the threshold parameter "number of cases" should be set to a rather higher value. The most preferable setting is the third one of strategy C.

Only if the workspace limitations become a real problem, other strategies should be considered. Strategy A provides results that are concerning the number of retrieved adaptable cases nearly as good as for strategy C, but the gained workspace reduction is rather small. Keeping the more adaptable cases (strategy B) results in a small case base, but only few adaptable cases can be found.

## 6 Conclusions

In this paper, we have briefly presented the application of Case-based Reasoning in our antibiotic therapy adviser ICONS; we have compared two retrieval algorithms and have presented results for different prototype strategies and different settings.

Simple indexing works much faster than the Hash-Tree algorithm. However, as for simple indexing the retrieval time increases with the number of stored cases and the Hash-Tree algorithm does not depend on the number of stored cases, both algorithms should have equal retrieval times when few thousands of cases are stored.

The best strategy to find many adaptable cases is obviously to keep all cases in the case base. However, every stored case increases the storage and workspace amount of our system by approximately 1.7 KB, which might lead to performance problems for much bigger case bases than our test set of 75 cases for just one of more than 80 areas. The best settings for keeping all cases (strategy C) and for erasing cases without additional information (strategy A) are those where a rather higher value for the threshold parameter "number of cases" is used. This results in more retrieved adaptable cases.

## References

1. Smyth, B., McKenna, E.: Modelling the competence of cases-bases. In: Smyth, B., Cunningham, P. (eds.): *Advances in Case-Based Reasoning*, 4th European Workshop, EWCBR-98, Lecture Notes in Artificial Intelligence, Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 208-220
2. Macura, R., Macura, K.: MacRad: Radiology image resources with a case-based retrieval system. In: Aamodt, A., Veloso, M. (eds.): *Case-Based Reasoning Research and Development*, First International Conference, ICCBR-95, Lecture Notes in Artificial Intelligence, Vol. 1010. Springer-Verlag, Berlin Heidelberg New York (1995) 43-54
3. Schmidt, R., Pollwein, B., Gierl, L.: Experiences with Case-Based Reasoning Methods and Prototypes for Medical Knowledge-Based Systems. In: Horn, W., Shahar, Y., Lindberg, G., Andreassen, S., Wyatt, J. (eds.): *Artificial Intelligence in Medicine, AIMDM'99*, Lecture Notes in Artificial Intelligence, Vol. 1620. Springer-Verlag, Berlin Heidelberg New York Springer Berlin (1999) 124-132
4. Bellazzi, R., Montani, S., Portinale, L.: Retrieval in a Prototype-Based Case Library: A Case Study in Diabetes Therapy Revision. In: Smyth, B., Cunningham, P. (eds.): *4th European Workshop on Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 64-75



5. Bichindaritz, I.: From Cases to Classes: Focusing on Abstraction in Case-Based Reasoning. In: Burkhard, H.-D., Lenz, M. (eds.): 4th German Workshop on Case-Based Reasoning, Humboldt University Berlin (1996) 62-69
6. Stottler, R. H., Henke, A. L., King, J. A.: Rapid Retrieval Algorithms for Case-Based Reasoning. International Joint Conference on Artificial Intelligence 11 (1989) 233-237
7. Bueno-Cavanillas, A., Delgado-Rodriguez, M., Lopez-Luque, A., Schaffino-Cano, S., Galvez-Vargas, R.: Influence of nosocomial infection on mortality rate in an intensive care unit. *Crit Care Med* 22 (1994) 55-60
8. Smyth, B., Keane, M. T.: Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval. In: Richter, M. M., Wess, S., Althoff, K.-D., Maurer, F. (eds.): First European Workshop on Case-Based Reasoning, University of Kaiserslautern (1993) 76-81
9. Lenz, M., Auriol, E., Manago M.: Diagnosis and decision support. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S.: Case-Based Reasoning technology, from foundations to applications. Lecture Notes in Artificial Intelligence, Vol. 1400. Springer-Verlag, Berlin Heidelberg New York (1998) 51-90
10. Wess, S., Althoff, K.-D., Derwald, G.: Improving the retrieval step in Case-Based Reasoning. In: Richter, M.M., Wess, S., Althoff, K.-D., Maurer, F. (eds.): First European Workshop on Case-Based Reasoning, University of Kaiserslautern (1993) 83-88
11. Schank, R. C.: Dynamic Memory: A Theory of Learning in Computer and People. Cambridge University Press, New York (1982)
12. Bareiss, R.: Exemplar-based Knowledge Acquisition. Academic Press, San Diego (1989)
13. Rosch, E., Mervis, C. B.: Family resemblances: studies in the structure of categories. *Cognitive Psychologie* 7 (1975) 573-605

# A Product Customization Module Based on Adaptation Operators for CBR Systems in E-Commerce Environments

Sascha Schmitt, Rainer Maximini, Gerhard Landeck, and Jörg Hohwiler

University of Kaiserslautern, Department of Computer Science  
Artificial Intelligence – Knowledge-Based Systems Group  
67653 Kaiserslautern, Germany  
{sschmitt,maximini,landeck,hohwille}@informatik.uni-kl.de

**Abstract.** Existing electronic shops based on CBR technology allow customers to search for adequate products by only specifying the attributes for the products in a fuzzy way. Unfortunately, most electronic shops do not further support customers after the retrieval step. However, especially configurable products could be customized at this stage. Based on the approach of interactive adaptation operators, we present a flexible system architecture for a customization module which can be easily integrated in electronic shops. Our approach of a modular adaptation concept is implemented and currently tested within the ESPRIT project **WEBSELL**<sup>1</sup>

## 1 Introduction

CBR technology becomes more and more important in e-Commerce scenarios, especially in online shops on the Internet or CD-ROM catalogs. CBR systems can be used to support electronic sales intelligently, which can be seen by the growing number of CBR applications on the Internet (e.g., real estates<sup>2</sup>, last minute flights<sup>3</sup>, accommodation booking<sup>4</sup>). Currently, it is possible for customers to select a product from a list of alternatives most appropriate to their requirements and demands. However, most of the systems do not offer further customer support after the retrieval step. As a consequence, such applications can only handle products which have to be chosen in a whole. But, after the retrieval step the sales system should offer an adaptation step like described by the classical CBR cycle [1]. Examples of appropriate products for adaptation are technical

---

<sup>1</sup> Project no. 27068. Partners: tec:inno GmbH (prime contractor, Germany), Adwired AG (Switzerland), Interactive Multimedia Systems (Ireland), IWT Magazin Verlags GmbH (Germany), Trinity College Dublin (Ireland), and the University of Kaiserslautern (Germany).

<sup>2</sup> <http://www.hookemacdonald.ie/>

<sup>3</sup> <http://www.reiseboerse.com/>

<sup>4</sup> <http://www.mueritz.de/mofruk3.htm>

equipment like computers [13], designs for electrical engineering [8], travels, or financial services.

A sales system providing intelligent sales support for products that can be modified to some degree has to offer the possibility to its users to customize the retrieved product, i.e. to adapt this product to the customer's wishes [7]. CBR research offers several methods for general case adaptation [5], [11], [12], but for product customization in electronic shops one has to distinguish carefully between these approaches because of different advantages and shortcomings.

The paper at hand describes a client/server system architecture which implements an interactive approach to operator-based adaptation [7]. The system works for both, business-to-business as well as business-to-consumer applications. To provide maximal flexibility, the adaptation system communicates with its controlling system, i.e. the retrieval system, via XML protocols. Our customization module has been implemented in Java. Besides, a client interface exists to let the customer communicate with the customization system. As a test environment for our implemented module serves the system architecture established by the ESPRIT project **WEBSELL**. The major outcome of this project is a toolkit for intelligent sales assistants for the WWW<sup>5</sup>. Because of its design, is it possible to take the customization module as an add-on to the toolkit. In **WEBSELL**, the retrieval system **ORENCE** [10] is used, a Java-based follow-up of CBR-Works [9]. Our customization module adapts completely to the open architecture of **ORENCE** which consists of single so-called services. For that reason, our module can be considered one of these services, too.

This work can be seen as the continuation of the approach described in [7]. In Chapter 2, we summarize briefly the idea of adaptation operators and we will point out which requirements the customization component had to fulfill. Chapter 3, discusses the architectural structure of the customization system. In Chapter 4 we present an example for a domain to which we applied our approach. We end with a discussion in Chapter 5.

## 2 Basic Requirements for the Customization Module

The previous chapter introduced briefly the idea of a flexible customization module or service that can be easily integrated in e-Commerce applications. As our module is based on the theory of adaptation operators as described in detail by [7], we start giving a short overview of the background of the operator approach before we present the technical, architectural issues of the customization module.

### 2.1 General Considerations about the Customization Process in e-Commerce

In e-Commerce applications a huge number of different product types exists. It is nearly impossible to formulate general requirements that hold for the sales of

<sup>5</sup> More information to be found under <http://www.wagr.informatik.uni-kl.de/~websell>

every kind of product. We derived requirements from general observations which we made by evaluating applications in the domains of vacations, cars, computers, and more sophisticated technical equipment like described in Chapter 4.

**Main Requirements from the Customer’s and the Shop-Builder’s Point of View.** The most important requirement is that the control of the customization of the product should always stay with the customer. Full automation of customization is not desired. The task of the system is to guide the customer through the process of customization. It has to check all possibilities for adapting the product and offering these to the user. A shop-builder wishes a flexible means of representation of the diverse possibilities of product customization and an application that serves both kinds of customers, business-to-business as well as business-to-consumer.

**Customization Based on a Transformational Adaptation Approach.** There are two major approaches for adaptation which can be distinguished, transformational and generative adaptation [3], [4]. The operator approach is based on a transformational methodology. Transformational adaptation modifies directly a retrieved “old” solution or case to adapt it to the new problem situation.

## 2.2 Product Customization by Adaptation Operators

The essential idea behind our customization approach is to divide the customization process into distinct steps. In each step, the product will be treated by an atomic unit of change. The customization system offers all possible changes to the product in the current situation and the customer decides which of these changes will be processed. To model these units of changes, the concept of customization operators was developed (see fig. 1). Such an adaptation operator contains the whole customization knowledge for this unit of change and it has to be defined by the builder of the electronic shop.



Fig. 1. Customization Operator

The first operator chosen and applied by the customer transforms the retrieved product into an intermediate product. Sequentially, the customer will apply more operators on the intermediate products until his target product is reached (see fig. 2).



is applicable is part of its definition and this must all be defined by the shop-builder. The kinds of domains to which the operator-based approach holds is limited. In a very complex domain the precondition for applicability of an operator would grow to dimensions which cannot be managed. In general, we can distinguish between three categories of products considering their ability to be customized:

- unchangeable products, e.g., integrated circuits, books, etc.
- products with few changeable features, e.g., vacations, houses, etc.
- products with many changeable features, e.g., computers, cars, design tasks, etc.

For the first category no adaptation is possible at all. The customization operator approach only works well for configurable products with few possibilities for modifications, and with low interaction among their features. The last category can only be supported by the operators up to a certain degree.

### 3 The Architecture of the Customization System

Our primary goal was to design the customization system very independent, flexible, scalable, and extensible.

To fulfill these requirements, the system itself is independent of any CBR-toolkit. The customization module could even work equally well with a standard retrieval system. Our customization system just customizes a given product controlled by a user. Nevertheless, in our scenario the product may be received from a CBR retrieval component but it is also possible to start with an empty product and let the user add components until he is satisfied. The system is divided in several submodules, which communicate by sending XML code via TCP/IP.

This section gives an overview of these parts and communication protocols.

#### 3.1 Client/Server Architecture

Figure 3 shows the client-server-based architecture of the whole customization system.

E-Commerce applications have the problem in common that their number of users is unknown at startup. Therefore, it is very important that the system is scalable. A client/server architecture can solve this problem.

On the client-side there is a Java Applet, which is able to run in a standard WWW browser. This is the user interface and only includes communication and presentation.

On the server-side we have two Internet server modules: The first one, the controlling service, allows the connection to external systems. This can be a complete CBR system itself or just a translator to a retrieval system. The only requirement is that it sends and receives the product to the client in a specified

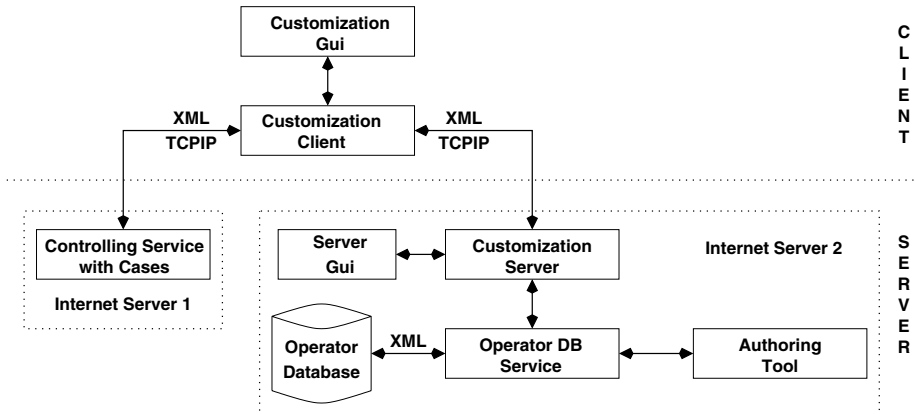


Fig. 3. Overview of the Client/Server Architecture

XML language. In principle, the customization system can be tailored to other e-Commerce toolkits just by replacing this server module.

The second server contains the main functionality. Via a simple GUI the shop builder can load the model and operators the customization server should work on. Once the server is started, it can answer queries from a client using another XML protocol.

Those three parts can run on different machines and it is possible that the server is also distributed over several machines. A typical customization process would perform the following steps:

1. After the customization Applet is loaded, the customization client initializes a connection to the controlling service and asks for the product to customize. The controlling service identifies the client and sends it the product.
2. The customization client sends the product to the customization server and asks for possible operators.
3. The customization server receives this product, checks all the operator conditions and sends the list of applicable operators.
4. The client presents these operators to the user who selects one of them and a corresponding parameter. The user selection is sent to the customization server, which customizes the product and checks all operators again. This step will be repeated, until the user terminates the loop by accepting or rejecting the product.
5. After the customization process, the final product is sent to the controlling service, which may take control of the e-Commerce application again.

### 3.2 Communication between the Modules

During language design, special attention was paid to the following qualities:

- Each submodule has its own language, which fulfills a special task.
- Each submodule can be easily replaced by another one, which just has to use the same language.
- The languages reuse specifications and parts of the ORENGE communication language.

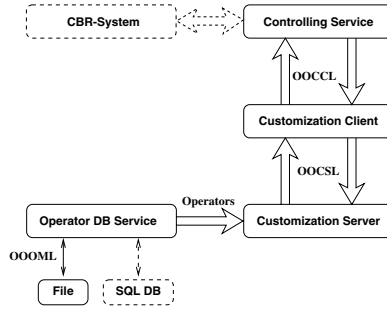


**Fig. 4.** Structural view of a typical communication example

Each component uses its own language, which makes reuse in other applications very easy. In our architecture these are three XML languages for the communication (see figure 4).

- **OOOML** is the **ORENGE Operator Object Meta Language**, which is used to store the operator definition in the file system or database. It only includes the definition data and no additional functionality.
- **OOCSL** (**ORENGE Operator Customization Service Language**) manages the communication between clients and the customization server. The client sends a query containing a description of the product and possibly the names of operator and parameter to customize this product. The server returns the maybe modified product and a list of the applicable operators with their parameters as result. After each query-result cycle the connection is closed. Because this protocol causes the only network traffic during the customisazion process it was designed very efficient.
- **OOOCL** is the **ORENGE Operator Customization Client Language**, which implements the communication between the controlling service and the customization client. It is used only at the beginning and at end of the customization process to transfer the product. The query-result structure is similar to **OOCSL**.

### 3.3 Implementation of the Operators

The operator concept is implemented in Java. Figure 5 shows the relations between the main classes.



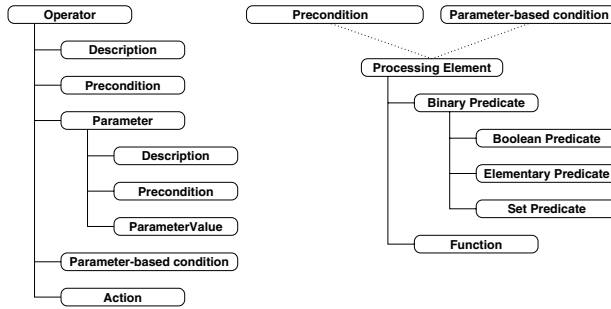


**Fig. 5.** Class Relations of the Operators

An operator, as an instance of the operator class, consists of five main parts:

- a description,
- several preconditions,
- optional parameters,
- several parameter-based conditions,
- several actions.

The condition classes include a processing element class, which bases on condition and function classes.

For the user it might seem very suspicious if after he had chosen an operator the system would tell him that this operator was not applicable or there was a problem, e.g., a division by zero. To avoid these circumstances there are preconditions which also do all complex calculations before operators are offered to the user. Each operator has global preconditions which are checked first. If they fail, the operator will be currently invalid. In the next step, the preconditions of the parameters of all remaining operators are tested in the same way. So, only applicable operators and corresponding parameters are presented to the user. After the user has selected, the parameter-based conditions are checked. They are necessary because the actions are reduced to simple assignments and some calculations, e.g., for the price, may depend on the selected parameter value. Finally, the actions can adapt the product accurately.

With this concept for an implementation, it is possible to develop very powerful conditions, because they can include conditions again. Additionally, it is very intuitive to create the operators, because they are designed to build them in a natural way.

### 3.4 Integration in a CBR System: Example ORENCE

Derived from the requirements in **WEBSELL**, the customization system is designed to work together with **ORENCE**. The **ORENCE** architecture is modular,

consisting of so-called *services*; each service implements one special functionality. As the customization system realizes such a service, it is possible to deliver it as an optional add-on.

## 4 Example for an Application Domain: Customization of Electro-Mechanical Components

As a testbed for our customization module, we have chosen an electronic product catalog, which is based on CBR technology. The catalog contains thousands of electro-mechanical components and devices like floating and magnetic switches, level controls, liquid level indicators, moisture detectors for cooling ceilings, etc.

### 4.1 The Electronic Catalog Based on CBR Technology

The electronic catalog was the major outcome of SMARTSELL<sup>6</sup>, another ES-PRIT project. It contains the products of the German company "Jola Spezialschalter K. Mattil & Co." In general, customers of Jola are neither very knowledgeable about the product variety nor do they know which kind of product is appropriate to their problem. The use of these very special technical parts in industry is underlying specific constraints. Even if a product has been found for the customer's purposes, s/he might have further specifications like length limitations or there is a need for accessories. For some components, the latter are even indispensable.

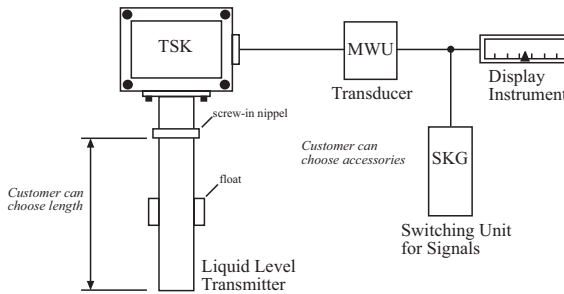


**Fig. 6.** Example for a customization situation

Figure 6 shows an example for a customization situation in the Jola domain. Suppose the retrieval system has returned a liquid level transmitter. For these types of switches, the customer has several possibilities for modifications. S/he

<sup>6</sup> Project no. 28835. Partners: Jola Spezialschalter K. Mattil & Co. (prime contractor, Germany), tec:inno GmbH (Germany), and Interactive Multimedia Systems (Ireland)

can adapt the length of the shaft, the diameter of the screw-in nipple, or the type of float of this switch. Furthermore, there are accessory components to choose like a transducer. Only depending on the latter component, further components can be chosen, e.g., a switching unit for signals and/or a display instrument. Figure 7 shows the customization module integrated in the Jola domain (up to now available in German only).

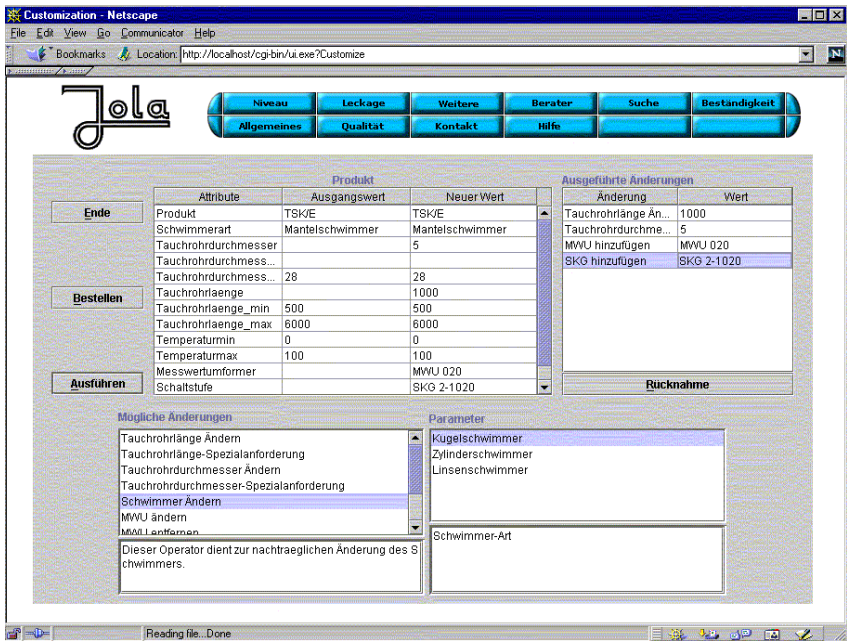


Fig. 7. Graphical user interface to the customization system of the Jola domain

With the help of our customization service as an add-on to the electronic catalog, it is possible to guide the customer, not being familiar with all the modification possibilities and constraints, to his or her target product.

## 4.2 Leaving Consistency Checking for More Flexibility

The Jola application elicited another interesting aspect. For us, it was always necessary to guarantee product consistency, i.e. the user of an operator could not enter parameters off the limits. Nevertheless, working on the Jola application, we found out that this might make sense, because they also produce special designs if required. Our operator concept can handle this, too. This is just a question of modelling. Of course, the customer should be informed by the user interface that such special designs cannot be guaranteed and may take more time

to manufacture. Such decisions can not be automated and must be determined by the sales personnel.

## 5 Discussion

To end this paper, we comment on the state of implementation of the described customization module. We will try to compare our approach to related ones and we will give a brief overview of work to be done in the future.

### 5.1 State of Implementation

The customization module described in this paper is fully implemented in Java and can be used as an add-on to an electronic shop. A showcase application to customize vacations had already been implemented. Currently, we are working on applying our adaptation operators to selected parts of the whole Jola product line of electro-mechanical components described in Chapter 4. At present, the CD-ROM catalog only exists as a prototypical implementation. By late Spring 2000, the CD-ROM catalog is expected to be ready for selling as well as an online version accessible in the Internet.

### 5.2 Related Work

Most of the discussion of the related work has already been done in [7]. An interesting example to add to the list of the visited online shops in [7] is the Subaru Car customization tool<sup>7</sup>. The customer entering the shop first selects a base model. Then, there is the possibility to further adapt this base product. Operators to modify exterior color, accessories, wheels, etc. are offered. The system checks consistency and supports the retraction of decisions. E.g., if the color 'red' has been chosen, the spoiler can only be chosen in red, too. If the color is changed to 'green' and a red spoiler had been chosen before the adaptation system takes away the spoiler but does not change it to 'green'. A big difference to our approach is that the Subaru system even offers possibilities, which change the product in an inconsistent way. In that case, the customer cannot select the change but will be displayed a more or less enlightening explanation. Their system does not offer operators with parameters. Furthermore, it would be desirable to have a directly displayed comparison between the base product chosen at the beginning and the modified product from the adaptation process. A nice but very limited feature is the picture of the selected car which changes color with the customer's choice of exterior color (cf. next section).

The system does not support a retrieval, i.e. the customer only has the possibility to further modify a base model which he selects from a given list. That means, customers cannot enter requirements on the vehicle and let the system

---

<sup>7</sup> <http://buildyourown.subaru.com/>

find out the most appropriate one. The Subaru online system is based on a Firepond<sup>8</sup> solution. Firepond offers e-business sales solutions for intelligent product recommendation.

### 5.3 Future Work

A couple of extensions to our customization module will be considered in future work. A nice enhancement could be the connection of the product with hypermedia information like a picture. One could imagine that the picture of the product would be regenerated with each modification. E.g., in our example domain this could be a schematic drawing of a level control switch with all its measures sized to the actual product settings.

## Acknowledgements

Funding for this research has been provided by the ESPRIT 4 Programme of the European Commission in the **WEBSSELL** project — Intelligent Sales Assistants for the World Wide Web, contract no. 27068.

## References

1. A. Aamodt, E. Plaza: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, in AICOM, Vol. 7, No. 1, 1994, pp. 39-59. 504
2. R. Bergmann, W. Wilke, I. Vollrath, S. Wess: Integrating General Knowledge with Object-Oriented Case Representation and Reasoning. In M. Lenz, H.-D. Burkhard (Eds.) 4th German Workshop: Case-Based Reasoning - System Development and Evaluation. 1996. 507
3. J. Carbonell: Derivational Analogy and Its Role in Problem Solving, Proceedings of the 3rd Annual National Conference on Artificial Intelligence AAAI-83, 1983, Morgan Kaufmann Publishers. 506
4. P. Cunningham, D. Finn, S. Slattery: Knowledge Engineering Requirements in Derivational Analogy, In Wess et al. (Eds.), Topics in Case-Based Reasoning Lecture Notes in AI, Springer, 1994. 506
5. K. Hanney, M. Keane, B. Smyth, P. Cunningham: Systems, tasks and adaptation knowledge: revealing some revealing dependencies, In A. Aamodt, M. Veloso (Eds.) Case-Based Reasoning Research and Development. pp.461-470. Springer, 1995. 505
6. Jola Product Catalog, Jola Spezialschalter K. Mattil & Co. Lambrecht (Pfalz), Germany, 1999.
7. S. Schmitt, R. Bergmann: Product Customization in an Electronic Commerce Environment using Adaptation Operators. 1999. Proceedings of the 7th German Workshop on Case-Based Reasoning, GWCBR'99, in Würzburg, March 3-5, 1999. 505, 514
8. I. Vollrath: Reuse of Complex Electronic Designs: Requirements Analysis for a CBR Application. In B. Smyth, P. Cunningham (Eds.): Advances in Case-Based Reasoning, Proceedings of the 4th European Workshop on CBR, EWCBR'98, Dublin. pp. 136-147. Springer, 1998. 505

---

<sup>8</sup> <http://www.firepond.com/>

9. S. Schulz: CBR-Works — A State-of-the-Art Shell for Case-Based Application Building. Proceedings of the 7th German Workshop on Case-Based Reasoning, GWCBR'99, Würzburg, Germany, March 3-5, 1999. 505
10. tec:inno GmbH. Das orange Framework. Internal Report, 1999. 505
11. A. Voss: Towards a methodology for case adaptation. In W. Wahlster (Ed.): 12th European Conference on Artificial Intelligence, ECAI'96. pp. 147-151. John Wiley and Sons, 1996. 505
12. W. Wilke, R. Bergmann: Techniques and Knowledge Used for Adaptation During Case-Based Problem Solving. In 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA-98, 1998. 505
13. W. Wilke: Knowledge Management for Intelligent Sales Support in Electronic Commerce, Dissertation, University of Kaiserslautern, Germany, 1998. 505

# Selecting and Comparing Multiple Cases to Maximise Result Quality after Adaptation in Case-Based Adaptive Scheduling

Steve Scott, Hugh Osborne, and Ron Simpson

School of Computing and Mathematics, University of Huddersfield  
Queensgate, Huddersfield, HD1 3DH, UK,  
{s.scott,h.r.osborne,r.s.simpson}@hud.ac.uk

**Abstract.** Recent Case-Based Reasoning research has begun to refocus attention on the problem of automatic adaptation of the retrieved case to give a fuller solution to the new problem. Such work has highlighted problems with the usefulness of similarity assessment of cases where adaptation is involved. As a response to this, methods of case selection are evolving that take adaptation into account. This current work looks more closely at the relationship between selection and adaptation. It considers experimental evidence considering adaptation of multiple cases for one problem. It argues that selection of the best case after adaptation will often make more efficient use of case knowledge than any attempt to pre-select a single case for adaptation.

## 1 Introduction

Case Based Reasoning has been used to develop many successful systems. All use the notion of case similarity and this area has been the focus of much research [2,6,8]. The paradigm has been most frequently taken up for use in Help Desk and similar products, where details of previous problems are retrieved by the use of similarity indexes accessed using matching features extracted from the new problem (for examples refer to [14]). The retrieved cases yield a set of similarity questions, the answers to which are then used to home in further, until a single case is found that describes the problem and its solution. This is basically a retrieval only implementation of Case Based Reasoning.

Although some of the classic Case-Based Reasoning systems such as Clavier involved adaptation, systems where the case is also adapted automatically to provide a solution to the new problem are less common. Perhaps for this reason, there appears to have been less detailed research into the efficiency, either theoretical or practical, of such systems (but see [1,5,9]).

The authors have been investigating Case-Based Reasoning for use in solving Nurse Rostering problems. This is a problem where adaptation is necessary. Several papers have been produced on different aspects of this problem and its solution using

Case-Based Reasoning [10,11,12,13]. The latest of these papers is under consideration at the moment and summarises work on understanding the behaviour of similarity measures where adaptation is involved. Since that paper was written new experiments have been done that extend the study to look at adaptation in the same way. Because these current experiments add to the previous work, some of the background material has been reproduced here.

In proposing "Adaptation Guided Retrieval", Smyth and Keane have identified the need for consideration of the effects of adaptation when choosing cases[9]. They argue that similarity is not enough to select a case that can be adapted "easily" to build a new solution. They propose selecting a subset of cases by use of similarity and then selecting a final case to be adapted by using adaptation knowledge in various ways to eliminate cases that cannot be adapted easily.

Amongst the experiments backing up our paper are some that consider the ideas of Smyth and Keane and which add further to the understanding of the effects of adaptation in building Case Based Reasoning systems. We suggest that the effects of adaptation on cases cannot be completely predicted at retrieval time, even using a broad interpretation of Adaptation Guided Retrieval. This leads to the further suggestion that, rather than homing in on one particular case, multiple cases should be selected, adapted and tested for the quality of their results. This strategy is designed to increase the chances of finding a solution as near to the optimum as possible.

This paper argues that, while Case-Based Reasoning systems are often expected to reduce the chances of obtaining optimality in order to increase the chances of solving the problem, it is still desirable to maximise the chances of achieving or approaching optimality by keeping options open as long as possible. The paper discusses the background and theory behind the current experiments, presents the results in more detail and concludes by discussing the implications.

## 2 The Experiments

Looking at our early results with a Case-Based Rostering solution, it became clear that result quality was dependent on the interaction of three variables with the current problem. These variables were:

1. The value of the cases in the case-base in relation to the current problem,
2. The effectiveness of the case selection mechanism in finding the best case for adaptation,
3. and, finally, the effect of the adaptation mechanism on the case selected.

These three are not independent variables. As outlined in the introduction, we were faced with the fact that for any given case-base and any given problem, there might be a range of possible case selection mechanisms. Choosing between these mechanisms was not easy because the quality of the final result depended on the effect of a given adaptation mechanism on the cases selected. This raised the question of how we could tell which was contributing to the quality of the result; the selection mechanism or the adaptation effectiveness.

There is not a large body of published work on the theoretical understanding of roosting problems. In response to this we first tried some experiments on the



Travelling Salesman Problem [12], because this is a very closely investigated domain. Here we looked at the use of Case-Based Reasoning and, with some additions, confirmed the conclusions of Smyth and Cunningham[9] that something is needed beyond similarity in selecting cases where adaptation is involved. In another set of investigations we looked at the use of genetic algorithms as a form of adaptation, using the case-base as the seed source of initial solution vectors. Here we found that random selection of cases from the case-base could provide viable solutions and again confirmed the work of others [13] in concluding that a case-based GA seed was better than a totally random one. This also supports our own rostering work that indicates that there is real value in using cases and adapting them even if selection efficiency is hard to prove (see [10,11]).

Our work in the Travelling Salesman domain clarified several aspects, including:

1. the value of cases in solving problems,
2. confirmation that the same solution could be reached from different cases because of the adaptation,
3. the value of an objective measure in assessing the quality of solutions,
4. confirmation that inefficient cases on one problem could provide efficient solutions to new problems after adaptation,
5. strengthening of the feeling that there was no obvious way to pick the case that would yield the best final result after adaptation.

Details of these investigations can be found in [12].

It was decided to investigate the behaviour of and relationship between similarity and adaptation further. To do this, a prototype system has been built to solve simple nurse rostering problems. There were two simple adaptation methods in the prototype that could be used in a variety of combinations. The quality of results is tested using an objective measurement function. A fixed case-base and a fixed set of problems were used for the tests. Several versions of the basic prototype were used to test different aspects of selection and adaptation behaviour:

1. First no selection was present so that all cases were used. Each case was adapted by choosing either of the two adaptation methods, as required, for each sub problem encountered needing adaptation. This allowed us to find the best case in the case base using this adaptation protocol.
2. The next version of the prototype took a subset of the case-base, using a range of selection mechanisms, and adapted each chosen case using the same adaptation protocol as above. One of the selection mechanisms used was an attempt to implement Adaptation Guided Retrieval in this domain. We were then able to compare the results of selection tactics with the result produced by the best case from the whole case-base.
3. Another version of the prototype repeated the tests in version two, but this time correction of all sub problems was attempted using the first adaptation method and then any remaining problems were subjected to the second adaptation method.
4. The final version was the same as version three, but with the two adaptation methods reversed.

A series of problem situations were created using different numbers of nurses to be rostered over a fourteen day period, with three shifts to be covered each day. On each

problem a different number of nurses were required to cover each of the three shifts on each of the days. In addition each of the nurses had to be allocated a full shift rota. If all the work shifts were nights this means working eight of the fourteen days. If a mix of night and day shifts was involved then ten days needed work shifts to be allocated. In pure vector permutation terms, these are reasonably large problems. To give some idea of the potential size of the search space it is worth considering the range of problem sizes covered in the runs explained here. In the smallest problem there were 10 nurses to roster, which means that each nurse needs to have 14 daily shifts allocated from one of four values (representing early, late, night or off shifts). This means that there are  $140^4$  possible solutions. The largest problems contained 25 nurses, with  $350^4$  possible solutions.

The case base was seeded with the aim of eliminating any bias towards a particular selection mechanism. To achieve this, it was populated with a range of cases produced by solving a completely different problem set. These different problems were solved using a range of different case-base combinations. In working through this different problem set, all cases in each case-base were used to solve each problem and the results were stored in the case-base and then an attempt was made to select from these results randomly. This resulted in a case-base with good similarity coverage on a range of problems. Once the case-base had been established, it was kept in the same state throughout all the experiments. As the same cases were being used on each problem, this reduced the variables being considered in any results.

An objective measure was designed to analyse solution quality. The definition of this measure does not affect the overall conclusions and a range of such measures could be designed, depending on a particular view on solution quality. In this case a good quality solution was one that covered all the shifts required in the problem definition with a strongly weighted measure to penalise solutions moving further away from this ideal. In addition a number of other minor solution quality factors were included, with less strong effects on the overall measure.

The adaptation algorithm used consisted of two methods aimed at improving the objective measure score. Both of these methods attempt to fix shifts where the initial allocation, using the shift patterns from the case, causes the shift to have too few nurses for the level of cover required. The first method attempts to swap a nurse's shift where that nurse has a day off on the day of the shift to be fixed and on another day is working on a shift where there are too many nurses. The second mechanism attempts a similar swap, but where the nurse is working a different shift on the same day as the shift to be fixed and that shift has too many nurses. In the first of the experiments these two methods were both available for the solution of each problem shift. This static form of adaptation was used to ensure that all differences in results could be attributed to the power of each selection mechanism in relation to the particular problem being considered. The last two experiments used one adaptation method only in attempting to solve each problem shift. The second method was then used to try to solve any remaining problems. This sequential use of the adaptation methods allowed us to look at the effects of different adaptation protocols on result quality. Each of the three adaptation protocols (interleaved, sequential order one and sequential order two) moves through the solution space differently, producing different results.

The version of the algorithm designed to look primarily at selection rather than adaptation was created selecting cases using five different measures. One of the measures used was a random selection of four cases from the case-base, but eliminating any duplicates. Within each of the non-random measures the cases were ordered and the top four cases in the list for each measure were selected. This meant that each problem was solved twenty times, though it is possible that different selection mechanisms did occasionally use the same case to solve any one problem. The objective measure score was calculated for each of the cases used. The non-random selection measures used were:

1. Cases were ordered by the absolute difference between the number of nurses being rostered in the case and the number of nurses being rostered in the problem. This was the similarity measure used in the very first algorithm that we developed.
2. Cases were ordered using the absolute difference between the total number of shifts required to be filled by the problem and the total number of shifts that were being filled when a given case was developed.
3. A further version of the shift similarity mechanism was used with a weighting factor in place to allow for the extra difficulties involved in scheduling night shifts. These are more complex because a full complement of nights reduces the working shifts that can be scheduled for a nurse and because there are in-built rules in the problem solver to ensure that a nurse does not work a night followed by a day-time shift.
4. An attempt to pattern match directly between the shifts allocated in the case and the shifts to be allocated in the problem. This has some similarities to 1. and 2. above but makes an attempt to predict cases that will be easy to adapt. This is explained further below.

As mentioned earlier, Smyth and Keane have proposed the use of adaptation knowledge to provide a better case selection mechanism where adaptation is involved [9]. Their work emphasises the need to find cases that are "easier" to adapt. In the domain in which they were working this made sense and it was also easy to see how it could be applied. The broader idea of guiding selection on the basis of adaptation knowledge is interesting beyond the particular circumstances set out by Smyth and Keane. We wanted to include similar selection mechanisms in our own experiments. However, it was not easy to see how this would apply in the nurse rostering domain. In addition all our past work led us to believe that there is a difference between a case that is easy to adapt and one that produces a high quality result. In the Travelling Salesman domain, for instance, it is possible to have a case that looks easy to adapt because it contains a high level of similarity in terms of towns covered, but which after adaptation produces a very inefficient route. In our attempt to produce such an adaptation guided retrieval mechanism, we reasoned that a case where the shift patterns allocated closely matched the shift patterns apparently required by the new problem would be easier to adapt. This was measured by choosing patterns that minimised the total absolute difference between nurses allocated to each shift within the pattern and the nurses required for that shift within the problem. Minimising this figure means that less moves from one shift to another will be required in adapting the raw solution to match the problem as closely as possible. This is different to the first two measures, which attempt to match the target

in the original problem with the target in the new problem. The effectiveness of this measure will be considered further after a discussion of the results obtained from the experiments.

### 3 The Experimental Results

The previous paper [13] produced by the authors contained an analysis of the results from twenty runs, each run based on a different problem. There were five groups of results, based on the five case selection mechanisms, and within each group there were four columns representing the results from selecting four separate cases within that group for each problem. The results were calculated by taking the objective score for that selection and subtracting the best objective score obtained from the whole case-base on that particular problem. This means that the lower the value shown the better the result was. No one selection mechanism could be guaranteed to give the best result on all twenty runs. Indeed no mechanism even approaches such a performance, with random selection often achieving better results than the problem structured selection mechanisms. Some other selections, however, were good on average rather than in terms of achieving the best individual results. Once again this seemed to highlight the fact that no one selection method is really the best in all situations.

General consideration of the ineffectiveness of the various selection mechanisms might suggest that random selection would be just as effective. This would have the advantage of reducing the knowledge acquisition requirement involved in setting up any heuristic selection mechanisms. In addition random selection will not suffer from changes in efficiency due to changing domain circumstances. However, random selection alone would not produce good results overall even though most often of all the individual case selection mechanisms used here it might produce the best result. The selection mechanisms were split into four heuristic groups, with four selections within each group. Where the mean performance of the four heuristic groups varied between 153 and 317.5, taking the first selection in each of the four heuristic groups and calculating their mean gave a figure of 122.25. This means again that use of more than one selection mechanism will almost certainly out-perform the use of one alone.

Closer inspection of the full set of results revealed that individual selection mechanisms vary wildly on their effectiveness from run to run. Similarly some runs have a much lower spread of results than others, these presumably being problems where there are more possible solutions at or near the optimum available from the given case-base. On several runs, none of the selections quite matches the best from the entire case-base, although in all cases the difference is small. This further highlights the difficulties of case-selection, where even multiple selection techniques cannot guarantee to find the best case in the case-base for a given problem and a given adaptation method.

The pattern match selection method, which was our attempt to implement the idea of adaptation guided retrieval in this domain, was no more successful than other retrieval mechanisms on our fixed adaptation tests. To look more closely at the effects of adaptation we also looked at the adaptation costs of each selection method on each

of the problems. We did this by counting the number of iterations required to correct shift problems. Figure 1 shows the plotted results of this calculation where each of the different lines represents a particular selection method. Where a particular line is high up on the Y axis scale, then that represents a high adaptation cost, a case that is not easy to adapt.

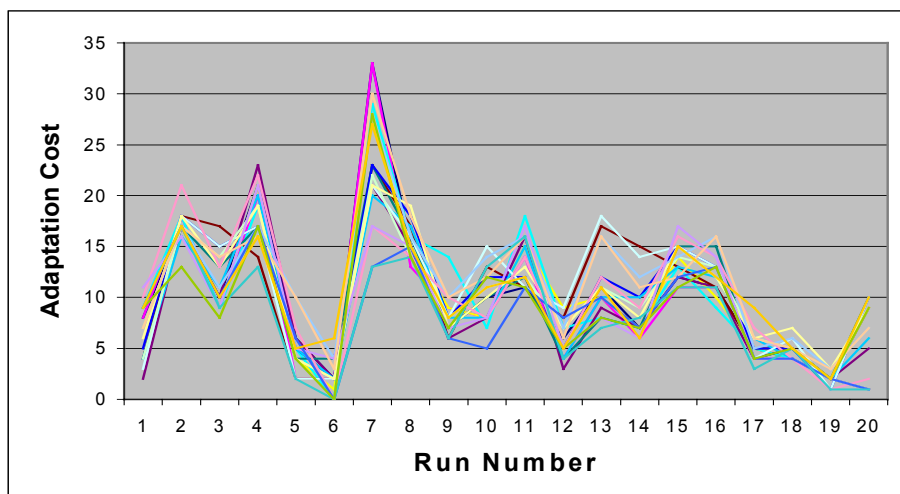


**Fig. 1.** Adaptation Costs

The adaptation costs are fairly obviously closely related to the particular problem being solved. Selection methods, however, vary significantly in their relative positions in the cost hierarchy on a problem by problem basis. This indicates that the same sort of conclusions might be drawn on adaptation costs as in other areas of the problem. What is not clear from Figure 1, however, is that the actual adaptation costs of the four selections attempting to select by adaptation costs are not always the lowest. This means that the selection method in this area has not proved a totally successful predictor of real adaptation costs. Consequently these experiments have not fully tested the value of adaptation guided retrieval.

The failure of the experiments to test the efficacy of a broad understanding of adaptation guided retrieval fully does not significantly affect the overall results or conclusions of this work. It was difficult to find an adaptation guided retrieval mechanism in this domain, perhaps because of imperfect adaptation knowledge. In addition the original observation, that speed and ease of adaptation does not equate to the effectiveness of adaptation in many problems, is still valid. This is backed up by the fact that the average adaptation costs of the AGR group of selections was significantly lower than the averages for the other groups. This means that, overall, this group was reducing adaptation cost and yet was not performing well on the measure of result quality. Additional experimentation is required to reach valid conclusions in this area.

Our experiments also looked at the effects of adaptation algorithms on solution quality so that a better understanding of the behaviour of selection/adaptation pairs

could be reached. This was done by running the same problems, with the same case-base, through the prototype using altered adaptation methods, as outlined earlier.

Once again the results of this were interesting. By changing the adaptation methods used each of the subset cases chosen could be seen to do one of three things:

1. produce exactly the same result as the original adaptation protocol,
2. produce a better result,
3. produce a worse result.

On different problems each selected case behaved in any of the three ways. This demonstrates that the behaviour of selection/adaptation pairs varies from one problem to the next. The behaviour of different adaptation methods tends to follow a problem based pattern. Thus, on any one problem a particular adaptation method might tend to give better results across the selection methods. On a different problem another of the adaptation methods might be superior. This was not universal though and there were some problems that favoured no particular method and similarly no method was always better on all the selected cases in a particular problem. Finally, in terms of the objective scores, in several problems the two new variants of the adaptation protocol managed to produce an objective score that was better than the best from the entire case-base using the original protocol. This could be balanced in the same problem by a score using another case that was far in excess of that using the original protocol. An example set of results are shown in Figure 2 and explained below.

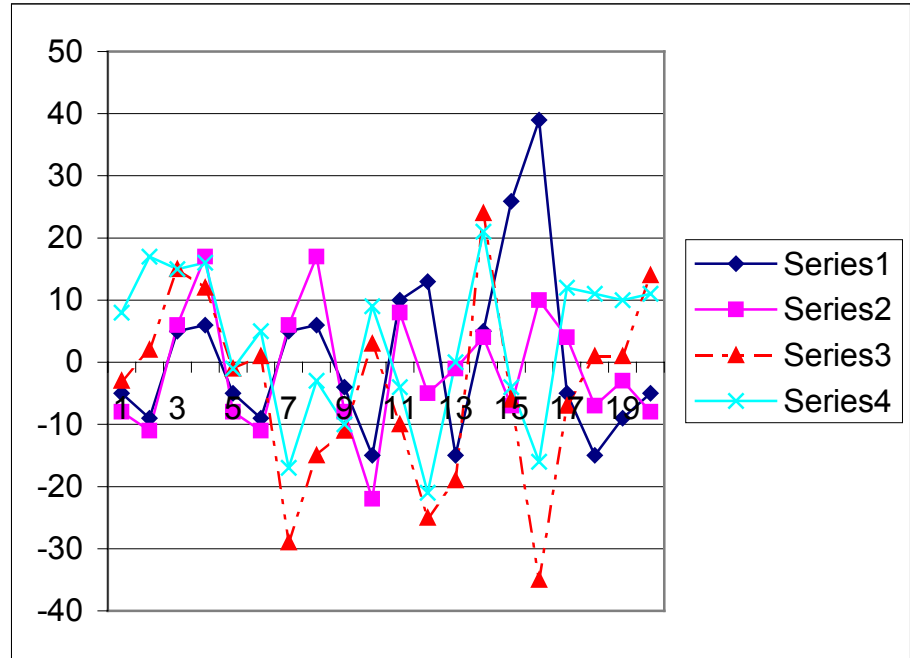


**Fig. 2.** Results from two runs showing adaptation differences

The results shown in figure 2 are from two runs on different problems. Series 1 shows the figure given by taking the adaptation method 1 results from the adaptation method

2 results. The two high point differences in this series have in fact been divided by one hundred in order not to skew the scale of the x-axis. Series 2 shows the figure given by taking the interleaved adaptation method results from the adaptation method 1 results on the same problem. Series 3 and Series 4 are the equivalent results for a different problem. The oscillating results give some indication of the complete variation of performance between problems and between selection and adaptation methods.

The complete variation on objective scores was generally not matched by similar behaviour with adaptation costs. While there was some variation between adaptation methods, this was not large. Very often the adaptation costs were the same for all three methods. At the same time adaptation costs would vary considerably between different cases on the same problem. On this set of problems and with these methods, adaptation cost tends primarily to be a feature of the case rather than the adaptation method.

## 4 Discussion

The results are interesting for anyone looking at case selection methods in systems where adaptation is involved. The final success of a system on a particular problem is dependent on:

1. the availability of a case in the case-base that can be adapted to solve the problem well,
2. the ability of the system to select the best case for adaptation,
3. the power of the adaptation algorithms to turn a given case into an effective solution.

To some extent a powerful case-base can make up for weaker adaptation algorithms and vice versa. Any weakness in either of these two will reduce the chances of the system achieving a result that is at or close to the optimum. However, the success of both of these parts is also dependent on the selection mechanism to feed the right case to the adaptation algorithm.

Even using a multiple case selection paradigm, it is possible to miss the best case for any one problem. However, in the tests that were conducted, this only happened once in terms of the performance on objective scores. This is in line with the general Case-Based Reasoning paradigm, which can sacrifice total optimality for solvability. Just as it is possible to miss the best case in the Case-Base, so it also possible for the best case not to be in the Case-Base. Similarly it is also possible that the adaptation algorithms used are not capable of turning the best case into the best result. In these matters our tests are not different to other Case-Based systems.

In addition to the overall results on the use of multiple selection mechanisms, there are some interesting results concerning the broad understanding of Adaptation Guided Retrieval. In general our attempt at implementing this in this domain did indeed reduce adaptation costs. However, this reduction in adaptation costs did not imply a similar level of result quality. In general the mechanism incorporating adaptation knowledge did not do well on our quality measure. The fact that adaptation methods can produce different quality results with each case/problem combination must also

cause us to ask whether keeping adaptation options open would produce better solutions than attempting to predict adaptation requirements at retrieval time. We conclude that more investigation into this area is needed

What our current work reveals is that the success of any given selection mechanism cannot be guaranteed across all problems. Use of multiple selection methods, followed by some form of solution grading and final selection of the best result increases the chances of success. Solution grading can be done using an objective measure, as in our experiments, or could be done by the users of any system produced using these techniques. Wherever solution grading is possible, it is to be recommended.

Similar conclusions must be drawn about adaptation algorithms. The most powerful adaptation algorithm possible has the effect of being able to turn any given case into any given solution. In such a situation the only value of the case-base is in reducing the time it might take to achieve the desired solution. A system with such an algorithm would potentially be using the entire search space and is unlikely to be implemented in all but the simplest situations.

Most case-based systems will have more limited adaptation algorithms and thus will rely more heavily on the power of the case-base and selection mechanisms to provide a seed that can be readily adapted into an efficient solution. Indeed problems with knowledge elicitation on adaptation and also with keeping adaptation current, have led to some investigation into automation of adaptation development [4]. This current work indicates that it is really only after the adaptation has been completed that the effectiveness of the selection can be judged. That effectiveness is dependent on the combination of the case and the adaptation (a case/adaptation pair). We have already pointed out that it is important not to be too pre-selective on cases, but the same is true of adaptation methods. The effectiveness of a given, limited adaptation algorithm is only finally obvious after application to a particular problem using a specific case. In the same way that it makes sense to use multiple selection mechanisms, it also makes sense to try multiple adaptation mechanisms, if this is feasible in a particular domain.

The possibility that more efficient systems can be produced by keeping both selection and adaptation methods open has other implications. Use of multiple methods should lessen the need for hand-crafted selection and adaptation methods. Work has been done on automatic generation of both similarity measures and adaptation protocols [7,4]. The retention of multiple options, with post-adaptation selection makes such automatic method generation techniques even more appealing.

Much as current work is looking at more formal analysis of methods of case-base maintenance, with the aim of a clearer understanding of which cases to retain and which to remove, so it may be possible to look at the more active elements of the system itself. Over time it may be possible to look at the performance of particular selection mechanisms and adaptation algorithms in order for the system to improve by favouring the more effective combinations. In this way Case-Based Reasoning systems could evolve not only in terms of an improving case-base but also by improvement of the system itself.



## 5 Conclusions

From an initial glance at the evidence, it is tempting at first to question the chances of cases, similarity measures and adaptation achieving efficient solutions. At the very least these current experiments question the ability of current Case-Based Reasoning methods to achieve such efficient solutions consistently where adaptation is involved. However, closer analysis leads to the conclusion that both case-bases and similarity measures have value in providing a basis for subsequent adaptation. What is clear is that singleton similarity measures or any singleton adaptation methods are likely to reduce the overall performance of the system. Care must be taken in selection of methods in both areas on a per-problem basis. Where possible, result quality should be graded to ensure that performance is understood.

Whatever the means of adaptation and grading of cases our work indicates that use of multiple cases should improve the chances of producing viable solutions. This emphasis on the use of multiple options when trying to solve a new problem seems to be in line with conclusions reached in the related field of machine learning, where classifier ensembles have been shown to be superior to singleton equivalents (see [16] for example).

## References

1. Bergmann, R., Wilke, W., "Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning", in Proceedings of ECAI98, Prade (Ed.) Wiley & Sons, 1998
2. Bergmann, R., Stahl, A., "Similarity Measures for Object Oriented Case Representations" in Advances in Case-Based Reasoning, Proceedings of the third European Workshop on Case-Based Reasoning, Lecture Notes in Artificial Intelligence, Springer Verlag 1996
3. Cunningham, P., Smyth, B., "On the use of CBR in Optimisation Problems such as the TSP", in Case-Based Reasoning Research and Development, Veloso, Aamodt (eds.), Lecture Notes in Artificial Intelligence, Springer Verlag 1995
4. Hanney, K., Keane, M. T., "Ease it by Learning from Cases", in Case-Based Reasoning Research and Development, Lecture Notes in Artificial Intelligence, Springer Verlag 1997
5. Janetko, D., Wess, S., Melis, E., "Goal Driven Similarity Assessment", Technical Report SR-92-05, Universität des Saarlandes, Saarbrücken, Germany, 1992
6. Kontkanen, P., Myllymäki, P., Silander, T., Tirri, H., "On Bayesian Case Matching", in Advances in Case-Based Reasoning, Smyth & Cunningham (eds.), Lecture Notes in Artificial Intelligence, Springer Verlag 1998
7. Osborne, H. R., Bridge, D., "Similarity Metrics: A Formal Unification of Cardinal and Non-Cardinal Similarity Measures", in Case-Based Reasoning Research and Development, Lecture Notes in Artificial Intelligence, Springer Verlag 1997
8. Richter, M. M., "Classification and Learning of similarity Measures" in Proceedings der Jahrestagung der Gesellschaft für Klassifikation, Opitz, Lausen Klar (ed.) Studies in Classification, Data Analysis and Knowledge Organisation, Springer Verlag, 1992
9. Smyth, B., Keane, M. T., "Adaptation Guided Retrieval: questioning the similarity assumption in reasoning", in Artificial Intelligence, Volume 102, pp 249-293, 1998
10. Scott, S. "Separating Constraint Dimensions in a Scheduling Problem to Reduce Search Space", In Proceedings of ECAI98, Prade (Ed.) Wiley & Sons, 1998

11. Scott, S., Simpson, R. M. "Case-Bases Incorporating Scheduling Constraint Dimensions: Experiences in Nurse Rostering", In *Advances in Case-Based Reasoning*, Smyth & Cunningham (eds.), Lecture Notes in Artificial Intelligence, Springer Verlag 1998
12. Scott, S., Osborne, H., Simpson, R., "Assessing Case Value in Case-Based Reasoning with Adaptation", In *Proceedings of the World Multiconference on Systems, Cybernetics and Informatics, IIS*, 1999
13. Scott, S., Osborne, H., Simpson, R., "The Case Selection Problem in Case-Based Reasoning - Making the best use of Case Knowledge for Adaptive Scheduling", under consideration.
14. Sushil, J. L., Li, G., "Augmenting Genetic Algorithms with Memory to Solve Travelling Salesman Problems", *Proceedings of the Joint Conference on Information Sciences*, Duke University, 1997
15. Watson, I., "Applying Case-Based Reasoning", Morgan Kaufmann, 1997
16. Dietterich, T. G., "Machine Learning: Four Current Directions" in *AI Magazine*. 18 (4), 97-136, 1997

# Author Index

- Aha, David W. .... 210, 322  
Althoff, Klaus-Dieter ..... 335
- Bandini, Stefania ..... 348  
Bellazzi, Riccardo ..... 467  
Bento, Carlos ..... 112  
Bergmann, Ralph ..... 273, 297  
Blanzieri, Enrico ..... 358  
Breslow, Leonard A. .... 210, 322  
Bresson, Benoit ..... 173  
Bridge, Derek ..... 74  
Burke, Robin ..... 370
- Cheetham, William ..... 15  
Craw, Susan ..... 136  
Cunningham, Pádraig ..... 49
- Díaz-Agudo, Belén ..... 37  
Doyle, Michelle ..... 49  
Dubois, Didier ..... 61
- Ebranati, Alessandro ..... 358
- Falkman, Göran ..... 380  
Ferguson, Alex ..... 74  
Ferrario, Maria Angela ..... 393  
Fox, Susan Eileen ..... 406  
Fuchs, Béatrice ..... 86
- Gierl, Lothar ..... 491  
Gomes, Paulo ..... 112  
González-Calero, Pedro A. .... 37  
Göker, Mehmet H. .... 99
- Hanft, Alexandre ..... 455  
Hoffmann, Achim ..... 260  
Hohwiler, Jörg ..... 504
- Hüllermeier, Eyke ..... 61, 124
- Iglezakis, Ioannis ..... 247
- Jappy, Pascal ..... 222  
Jarmulak, Jacek ..... 136  
Jorge, Alípio ..... 26
- Khan, Abdus Salam ..... 260  
Knight, Brian ..... 418  
Kontkanen, Petri ..... 148
- Lahtinen, Jussi ..... 148  
Landeck, Gerhard ..... 504  
Leake, David B. .... 161  
Li, Sheng ..... 3  
Lieber, Jean ..... 173  
Lopes, Alneu de Andrade ..... 26
- Manzoni, Sara ..... 348  
Maximini, Rainer ..... 504  
McGinty, Lorraine ..... 431  
McKenna, Elizabeth ..... 186  
McSherry, David ..... 198  
Mejasson, Patrick ..... 418  
Micarelli, Alessandro ..... 443  
Mileman, Tony ..... 418  
Mille, Alain ..... 86  
Minor, Mirjam ..... 455  
Montani, Stefania ..... 467  
Muñoz-Avila, Héctor ..... 210, 322  
Müller, Wolfgang ..... 335  
Myllymäki, Petri ..... 148
- Nau, Dana S. .... 210  
Neri, Alessandro ..... 443  
Nick, Markus ..... 335  
Nock, Richard ..... 222

- Osborne, Hugh ..... 517
- Perner, Petra ..... 479
- Petridis, Miltos ..... 418
- Portinale, Luigi ..... 234, 467
- Prade, Henri ..... 61
- Preddy, Keith ..... 418
- Reinartz, Thomas ..... 247
- Roth-Berghofer, Thomas ..... 247
- Rowe, Ray ..... 136
- Sansonetti, Giuseppe ..... 443
- Schmidt, Rainer ..... 491
- Schmitt, Sascha ..... 504
- Schumacher, Jürgen ..... 273
- Scott, Steve ..... 517
- Sebban, Marc ..... 222
- Shiu, Simon Chi Keung ..... 285
- Simpson, Ron ..... 517
- Smyth, Barry ..... 1, 186, 393, 431
- Snoek, Björn ..... 335
- Stahl, Armin ..... 297
- Stefanelli, Mario ..... 467
- Sun, Cai Hung ..... 285
- Thompson, Cynthia A. .... 99
- Tirri, Henry ..... 148
- Torasso, Pietro ..... 234
- Vollrath, Ivo ..... 309
- Wang, Xi Zhao ..... 285
- Weber, Rosina ..... 210, 322
- Wilson, David C. .... 161
- Yang, Qiang ..... 3
- Yeung, Daniel So ..... 285