

Lorraine McGinty  
David C. Wilson (Eds.)

LNAI 5650

# Case-Based Reasoning Research and Development

8th International Conference  
on Case-Based Reasoning, ICCBR 2009  
Seattle, WA, USA, July 2009, Proceedings

 Springer

Lecture Notes in Artificial Intelligence 5650

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Lorraine McGinty David C. Wilson (Eds.)

# Case-Based Reasoning Research and Development

8th International Conference  
on Case-Based Reasoning, ICCBR 2009  
Seattle, WA, USA, July 20-23, 2009  
Proceedings



Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada  
Jörg Siekmann, University of Saarland, Saarbrücken, Germany  
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Lorraine McGinty  
University College Dublin  
UCD School of Computer Science and Informatics  
Belfield, Dublin 4, Ireland  
E-mail: lorraine.mcginty@ucd.ie

David C. Wilson  
University of North Carolina at Charlotte  
College of Computing and Informatics  
Department of Software and Information Systems  
9201 University City Boulevard, Charlotte, NC 28223-0001, USA  
E-mail: davils@uncc.edu

Library of Congress Control Number: 2009929550

CR Subject Classification (1998): I.2, J.4, J.1, F.4.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743  
ISBN-10 3-642-02997-3 Springer Berlin Heidelberg New York  
ISBN-13 978-3-642-02997-4 Springer Berlin Heidelberg New York

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

springer.com

© Springer-Verlag Berlin Heidelberg 2009  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12717316 06/3180 5 4 3 2 1 0



# Preface

The International Conference on Case-Based Reasoning (ICCBR) is the pre-eminent international meeting on case-based reasoning (CBR). ICCBR 2009 (<http://www.iccbr.org/iccbr09/>) was the eighth in this series of biennial international conferences highlighting the most significant contributions to the field of CBR. The conference took place during July 20–23, 2009 in Seattle, Washington, USA. Previous ICCBR conferences have been held in Belfast, Northern Ireland, UK (2007), Chicago, USA (2005), Trondheim, Norway (2003), Vancouver, Canada (2001), Seon, Germany (1999), Providence, Rhode Island, USA (1997), and Sesimbra, Portugal (1995).

Day 1 of the conference hosted an Applications Track and introduced the first ICCBR Doctoral Consortium. The Applications Track focused on real-world examples of the use of CBR in industrial fielded applications and provided a cooperative forum for the cross-fertilization of ideas between academic and industry attendees. The Doctoral Consortium concentrated on providing a supportive atmosphere for early-stage CBR researchers. Participants were matched with mentors, who provided guidance on their research trajectory, as well as advice about writing and presenting their research.

Day 2 featured topical workshops on “Reasoning from Experiences on the Web,” “Case-Based Reasoning for Computer Games,” “Uncertainty, Knowledge Discovery and Similarity in Case-Based Reasoning,” and “CBR in the Health Sciences.” In addition, the second CBR Computer Cooking Contest was held following on from its success at the 9th European Conference on Case-Based Reasoning (ECCBR 2008) held in Trier, Germany. Students and research groups described their entries in terms of the application of case retrieval, adaptation, and combination methods for cooking recipes, and participated in an energetic live demonstrative competition that was enjoyed by all attendees of the conference.

Days 3 and 4 comprised presentations and posters on theoretical and applied CBR research, as well as invited talks from two distinguished researchers: Susan Craw, Director of the Research Institute for Innovation, DEsign And Sustainability (IDEAS) at the Robert Gordon University, and Edwina L. Rissland from the Department of Computer Science at the University of Massachusetts at Amherst. The presentations and posters covered a wide range of CBR topics of interest both to practitioners and researchers, including: textual CBR, reinforcement learning, biologically inspired CBR, Web search and CBR, case-based recommender systems, CBR tools and system design, CBR quality management, case library maintenance, case-based planning, case representation, case similarity and retrieval, multi-dimensional indexing, spatial prediction, case-based decision analysis, automatic case generation, temporal CBR, active learning, real-time CBR, games and simulations, legal reasoning, ambient intelligence, evaluation

methodology, case adaptation, hybrid AI systems, collective classification, video understanding, multi-agent collaborative systems, machine learning.

This volume comprises papers of all the presentations and posters. These 34 papers were selected from a total of 55 submissions; 17 papers for oral presentation and 17 papers for poster presentation. Each paper was reviewed by at least three Program Committee members and revised in line with the constructive feedback provided. The papers in this volume provide a representative snapshot of current CBR research. We have organized the proceedings into three categories: *Invited Talks* (2 short papers), *Theoretical/Methodological Research Papers* (25 papers), and *Applied Research Papers* (9 papers).

Many people participated in making ICCBR 2009 a success. Local Conference Chair Isabelle Bichindaritz, from the University of Washington at Tacoma, devoted a great deal of her time to organizing the local arrangements and liaising with our sponsors. Everything from the room arrangements to the conference banquet was handled by Isabelle, and for this we are very grateful. Also, special thanks to Bill Cheetham and Kareem Aggour our Applications Track Chairs, Susan Fox for organizing the Doctoral Consortium, and Mirjam Minor, David Leake, and Armin Stahl for arranging and over-seeing the second CBR Cooking Contest. We appreciate the sterling work of Workshop Coordinator Sarah Jane Delany and all the chairs of the respective workshops and their various committee members. Sincere thanks to the Program Committee and the additional reviewers for their co-operation during the paper review process. In addition, we would like to thank our invited speakers, all of the authors who submitted papers to the conference, and gratefully acknowledge the generous support of our sponsors.

Finally, we would like to express our gratitude to the previous ICCBR and ECCBR conference organizers who gave us very valued advice, and to Springer for its continued support with respect to publishing this series of conference proceedings.

July 2009

Lorraine McGinty  
David C. Wilson



## VIII Organization

Ralph Bergmann	University of Hildesheim, Germany
Enrico Blanzieri	University of Trento, Italy
Derek Bridge	University College Cork, Ireland
Robin Burke	DePaul University, Chicago, USA
Hans-Dieter Burkhard	Humboldt University Berlin, Germany
Susan Crow	Robert Gordon University, Scotland, UK
Pádraig Cunningham	University College Dublin, Ireland
Belen Diaz-Agudo	Complutense University of Madrid, Spain
Peter Funk	Mälardalen University, Sweden
Ashok Goel	Georgia Institute of Technology, USA
Mehmet H. Göker	PriceWaterhouseCoopers, USA
Andrew Golding	Lycos Inc., USA
Pedro A. González-Calero	Complutense University of Madrid, Spain
Christiane Gresse von Wangenheim	Universidade do Vale do Itajai, Brazil
Kalyan Moy Gupta	Knexus Research Corporation, USA
Eyke Hüllermeier	University of Marburg, Germany
Igor Jurisica	Ontario Cancer Institute, Canada
Deepak Khemani	IIT Madras, India
Luc Lamontagne	Université Laval, Canada
David Leake	Indiana University, USA
Ramon López de Mántaras	IIIA-CSIC, Spain
Michel Manago	kiolis, France
Cindy Marling	Ohio University, USA
David McSherry	University of Ulster, Northern Ireland, UK
Mirjam Minor	University of Trier, Germany
Héctor Muñoz-Avila	Lehigh University, USA
David Patterson	University of Ulster, Northern Ireland, UK
Petra Perner	Institute of Computer Vision and Applied CS, Germany
Enric Plaza	IIIA-CSIC, Spain
Luigi Portinale	University of Eastern Piedmont, Italy
Lisa S. Purvis	Xerox Corporation, NY, USA
Francesco Ricci	ITC-irst, Italy
Michael Richter	University of Calgary, Canada
Thomas Roth-Berghofer	DFKI, Germany
Rainer Schmidt	Universität Rostock, Germany
Barry Smyth	University College Dublin, Ireland
Raja Sooriamurthi	Carnegie Mellon University, USA
Armin Stahl	German Research Center for Artificial Intelligence (DFKI), Germany
Brigitte Trousse	INRIA Sophia Antipolis, France

Ian Watson	University of Auckland, New Zealand
Rosina Weber	Drexel University, USA
Nirmalie Wiratunga	Robert Gordon University, Scotland, UK
Qiang Yang	Hong Kong University of Science and Technology

## Additional Referees

Ibrahim Adeyanju	Nadezhda Govedarova	Babak Mougouie
Mobyen Uddin Ahmed	Sidath Gunawardena	Erik Olsson
Kerstin Bach	Alexandre Hanft	M.A. Raghunandan
Enrico Blanzieri	Chad Hogg	Meike Reichle
Alessio Bottrighi	Tor Gunnar Houeland	Niall Rooney
Sutanu Chakraborti	Stephen Lee-Urban	Ilya Waldstein
Mykola Galushka	Giorgio Leonardi	Ning Xiong

## Sponsoring Institutions

ICCBR 2009 was supported by The Boeing Company, The Defense Advanced Research Projects Agency/Information Processing Techniques Office (DARPA/IPTO), Empolis, The US Naval Research Laboratory, and Verdande Technology.

# Table of Contents

## Invited Talks

We're Wiser Together .....	1
<i>Susan Crow</i>	
Black Swans, Gray Cygnets and Other Rare Birds .....	6
<i>Edwina L. Rissland</i>	

## Theoretical/Methodological Research Papers

Case Retrieval Reuse Net (CR2N): An Architecture for Reuse of Textual Solutions .....	14
<i>Ibrahim Adeyanju, Nirmalie Wiratunga, Robert Lothian, Somayajulu Sripada, and Luc Lamontagne</i>	
Case-Based Reasoning in Transfer Learning .....	29
<i>David W. Aha, Matthew Molineaux, and Gita Sukthankar</i>	
Toward Modeling and Teaching Legal Case-Based Adaptation with Expert Examples .....	45
<i>Kevin Ashley, Collin Lynch, Niels Pinkwart, and Vincent Allevin</i>	
Opportunistic Adaptation Knowledge Discovery .....	60
<i>Fadi Badra, Amélie Cordier, and Jean Lieber</i>	
Improving Reinforcement Learning by Using Case Based Heuristics .....	75
<i>Reinaldo A.C. Bianchi, Raquel Ros, and Ramon Lopez de Mantaras</i>	
Dimensions of Case-Based Reasoner Quality Management .....	90
<i>Annett Bierer and Marcus Hofmann</i>	
Belief Merging-Based Case Combination .....	105
<i>Julien Cojan and Jean Lieber</i>	
Maintenance by a Committee of Experts: The MACE Approach to Case-Base Maintenance .....	120
<i>Lisa Cummins and Derek Bridge</i>	
The Good, the Bad and the Incorrectly Classified: Profiling Cases for Case-Base Editing .....	135
<i>Sarah Jane Delany</i>	
An Active Approach to Automatic Case Generation .....	150
<i>Michael W. Floyd and Babak Esfandiari</i>	

Four Heads Are Better than One: Combining Suggestions for Case Adaptation . . . . .	165
<i>David Leake and Joseph Kendall-Morwick</i>	
Adaptation versus Retrieval Trade-Off Revisited: An Analysis of Boundary Conditions . . . . .	180
<i>Stephen Lee-Urban and Héctor Muñoz-Avila</i>	
Boosting CBR Agents with Genetic Algorithms . . . . .	195
<i>Beatriz López, Carles Pous, Albert Pla, and Pablo Gay</i>	
Using Meta-reasoning to Improve the Performance of Case-Based Planning . . . . .	210
<i>Manish Mehta, Santiago Ontañón, and Ashwin Ram</i>	
Multi-level Abstractions and Multi-dimensional Retrieval of Cases with Time Series Features . . . . .	225
<i>Stefania Montani, Alessio Bottrighi, Giorgio Leonardi, Luigi Portinale, and Paolo Terenziani</i>	
On Similarity Measures Based on a Refinement Lattice . . . . .	240
<i>Santiago Ontañón and Enric Plaza</i>	
An Overview of the Deterministic Dynamic Associative Memory (DDAM) Model for Case Representation and Retrieval . . . . .	256
<i>Stefan Pantazi</i>	
Robust Measures of Complexity in TCBR . . . . .	270
<i>M.A. Raghunandan, Sutanu Chakraborti, and Deepak Khemani</i>	
S-Learning: A Model-Free, Case-Based Algorithm for Robot Learning and Control . . . . .	285
<i>Brandon Rohrer</i>	
Quality Enhancement Based on Reinforcement Learning and Feature Weighting for a Critiquing-Based Recommender . . . . .	298
<i>Maria Salamó, Sergio Escalera, and Petia Radeva</i>	
Abstraction in Knowledge-Rich Models for Case-Based Planning . . . . .	313
<i>Antonio A. Sánchez-Ruiz, Pedro A. González-Calero, and Belén Díaz-Agudo</i>	
A Scalable Noise Reduction Technique for Large Case-Based Systems . . . . .	328
<i>Nicola Segata, Enrico Blanzieri, and Pádraig Cunningham</i>	
Conceptual Neighborhoods for Retrieval in Case-Based Reasoning . . . . .	343
<i>Ben G. Weber and Michael Mateas</i>	
CBR Supports Decision Analysis with Uncertainty . . . . .	358
<i>Ning Xiong and Peter Funk</i>	

Constraint-Based Case-Based Planning Using Weighted MAX-SAT . . . . .	374
<i>Hankui Zhuo, Qiang Yang, and Lei Li</i>	

## Applied Research Papers

A Value Supplementation Method for Case Bases with Incomplete Information . . . . .	389
<i>Kerstin Bach, Meike Reichle, and Klaus-Dieter Althoff</i>	
Efficiently Implementing Episodic Memory . . . . .	403
<i>Nate Derbinsky and John E. Laird</i>	
Integration of a Methodology for Cluster-Based Retrieval in jColibri . . . . .	418
<i>Albert Fornells, Juan Antonio Recio-García, Belén Díaz-Agudo, Elisabet Golobardes, and Eduard Fornells</i>	
Case-Based Collective Inference for Maritime Object Classification . . . . .	434
<i>Kalyan Moy Gupta, David W. Aha, and Philip Moore</i>	
Case-Based Reasoning for Situation-Aware Ambient Intelligence: A Hospital Ward Evaluation Study . . . . .	450
<i>Anders Kofod-Petersen and Agnar Aamodt</i>	
Spatial Event Prediction by Combining Value Function Approximation and Case-Based Reasoning . . . . .	465
<i>Hua Li, Héctor Muñoz-Avila, Diane Bramsen, Chad Hogg, and Rafael Alonso</i>	
Case-Based Support for Forestry Decisions: How to See the Wood from the Trees . . . . .	479
<i>Conor Nugent, Derek Bridge, Glen Murphy, and Bernt-Håvard Øyen</i>	
A Case-Based Perspective on Social Web Search . . . . .	494
<i>Barry Smyth, Peter Briggs, Maurice Coyle, and Michael P. O'Mahony</i>	
Determining Root Causes of Drilling Problems by Combining Cases and General Knowledge . . . . .	509
<i>Samad Valipour Shokouhi, Agnar Aamodt, Pål Skalle, and Frode Sørmo</i>	
<b>Author Index</b> . . . . .	<b>525</b>



# We're Wiser Together

Susan Crow

Computing Technologies Centre  
Robert Gordon University, Aberdeen  
s.crow@rgu.ac.uk

**Abstract.** Case-Based Reasoning solves new problems by reusing solutions from individual experiences stored in the case base. This paper explores beyond the explicit knowledge captured as individual experiences for problem-solving. Instead, the collective knowledge of the case base provides additional implicit knowledge that may be exploited to improve traditional case-based reasoning systems through additional knowledge containers and to provide a new form of self-adaptive case-based reasoning. This paper presents a view that individual experiences are more fully utilised through the wisdom of collective memory.

## 1 Introduction

Case-Based Reasoning (CBR) solves new problems by retrieving similar problems and reusing the solutions [1,2]. The collection of previously solved problems forms the main knowledge source for a CBR system and this case base is at the heart of any CBR system.

The individual experiences captured in cases provide wisdom locally in the problem-solving space. However, the case base is a collection of individual experiences and this paper proposes that the implicit knowledge captured in a collection of cases offers greater potential. This paper explores how this knowledge can be discovered, extracted, understood and exploited to achieve greater wisdom than the sum of the individual parts.

## 2 Knowledge Containers

Richter's ICCBR-95 keynote proposed further CBR knowledge containers in addition to the case base: representation language, similarity knowledge and adaptation knowledge [3]. He notes that the containers' content is not fixed and that knowledge can be shifted between containers. Here, we explore how the collection of case knowledge enables the discovery of implicit representation, retrieval and adaptation knowledge that improves the performance of a CBR system.

### 2.1 Retrieval Knowledge

CBR retrieval often combines indexing to identify a relevant cases in the case base and a nearest neighbour retrieval to select the most similar cases for reuse. In this scenario retrieval knowledge includes the attributes selected for indexing

the case base and the relative weightings of attributes for nearest neighbour retrieval. We have used the case base as a source of training data to learn attribute selections and weightings [4]. A simple genetic algorithm (GA) wrapper approach represents the attribute selections and weights as binary- and real-valued genes. The GA fitness function is the accuracy from a leave-one-out testing on the case knowledge in which the retrieval is instrumented with the attribute selections and weights. This leave-one-out accuracy determines the population of most fit attribute selections and weights for the next GA iteration.

The learned retrieval knowledge improved the predictive accuracy for a demanding pharmaceutical application. Designing a tablet to deliver a new drug involves choosing ingredients for other components (filler, binder, disintegrant lubricant and surfactant) and their quantities to enable a viable tablet to be manufactured. Many of the components and quantities showed significant improvement. Predicting the filler and binder components is particularly important and improving retrieval alone was not sufficient. Learning retrieval knowledge was also effective when the formulation task changed and this semi-automated re-engineering of the knowledge was competitive with an expensive hand-crafting of rules to cope with the changes [5].

## 2.2 Adaptation Knowledge

Differences between pairs of cases that solve each other are a source of implicit knowledge about solution adaptations. Our adaptation learning for tablet formulation built on the leave-one-out testing employed for retrieval knowledge. Adaptation training examples were created from the contents and differences of the new excluded problem and its retrieved neighbours, and their solutions. Effective adaptation knowledge was learned using various decision tree and nearest neighbour algorithms [6].

But binder prediction is a particularly demanding task and remained elusive. Binder classification required a less coarse adaptation than simply suggesting a new binder because the choice of binder is a tradeoff between different properties of the drug and the other components. The solution was to create novel property-based adaptations and an ensemble approach to adaptation [7]. A substantial improvement was achieved for binder prediction; a 1-NN adaptation ensemble doubled the average binder accuracy from 34% to 74%!

A consensus approach to adaptation has also proved effective for a similarly challenging workflows application where balancing conflicting requirements is achieved by exploiting “wisdom of crowds” [8].

## 2.3 Representation Knowledge

In textual CBR the discovery of attributes is more refined than simply selection as in section 2.1. Variety in vocabulary usage and the absence of structure are interesting semantic challenges for textual case representation. Exploitation of document collections provides examples and evidence from which to identify keywords and phrases, to define and organise key concepts, and to index and retrieve cases that are stored as textual documents [9,10,11]. These approaches

have proved successful for reusing satellite anomaly reports and occupational therapy reports about SmartHouse solutions for independent living.

### 3 Meta-knowledge

The case base is often assumed to be representative of the problem-solving domain and so collections of cases provide meta-knowledge case knowledge.

#### 3.1 Models and Maintenance

Smyth & McKenna's competence model groups cases that solve, and are solved by, each other [12]. Competence groups map out the landscape of cases and how they are related to each other. Our complexity-based model focuses on interactions at class boundaries [13]. We gain insight into the global structure of the case base by constructing a profile of case complexities that indicates the level of redundancy and complexity in the case base, and the complexity score for individual cases identifies areas of regularity compared to more complex class boundaries. An understanding of relationships in the case knowledge makes explicit meta-knowledge implicit in the case base.

The resulting models of competence enable case acquisition tools that discover new cases [13], and case base editing tools that discover new cases or remove redundant or noisy cases [14,15].

#### 3.2 Agile CBR

This paper has focused on exploiting implicit knowledge in the case base for the acquisition and maintenance of the various CBR knowledge containers. This section proposes a new type of CBR that captures some of the key features of the *agile* approach now popular in software development. Agile CBR targets the CBR cycle rather than CBR knowledge, and considers how it can utilise the experience collection as well as the individual experiences in new ways to provide more sophisticated reasoning.

One approach to Agile CBR replaces the standard RETRIEVE-REUSE-REVISE cycle of traditional CBR with a RETRIEVE-VALIDATE-REVISE meta-cycle that utilises the standard CBR cycle in different ways at each stage [16]. Agile retrieve will reuse a retrieved experience from the case base, but not to solve the problem, instead to suggest a new part of an evolving solution; i.e. some next steps towards a solution. Agile validate will seek corroboration for these suggested partial solutions from the case base to approve and prioritise the suggestions. Agile revise will search the case base for refinements to the solution that reflect differences in the new problem. Thus the case base is used as a source of individual problem solving experiences, but also as a collection of cases that may help to confirm and/or revise parts of the solution.

Viewing the case base as general experiences that can be utilised in different ways is reminiscent of Goel's early research on Router for robot spatial reasoning [17], Laird's more recent Episodic Memory [18], the CBE (Case-Based Engineering) architecture that imitates a human engineer's targeted trial-and-error experiments

to discover the right inputs for given outputs, or to specify some inputs and outputs in order to discover others [19], and Goel's current research on Meta-CBR [20].

Agile CBR exploits the experiences in the case base in different ways both individually but also more generally as representing a problem-solving landscape. Thus Agile CBR transforms traditional CBR into a dynamic, knowledge-rich, self-organising, cooperative problem-solving. The need for more flexible, adaptive techniques is beginning to be recognised for designing processes and workflows [21]. Agile CBR aims to embed agility in the reasoning rather than in the case representation and similarity matching.

## 4 Conclusions

This paper has contrasted a case-based problem-solving from individual cases with a knowledge enriched CBR problem-solving where knowledge in other knowledge containers has been automatically acquired by exploiting the collective knowledge in the case collection. This has led to the prospect of self-adapting CBR where firstly the case base provides an understanding of the problem-solving landscape that enables self-adaptation of the case base, to self-adaptive case-based reasoning where Agile CBR itself determines the next piece of reasoning and the individual cases suggest partial solutions, but the case-base provides evidence to support validation and suggestions for revision.

CBR is synonymous with reasoning from experiences, but until recently has in practice meant reasoning from one or more individual experiences. Thus CBR equates to Wisdom from similar individuals. This paper has emphasised the added value from a collection of experiences. Thus reasoning from experiences goes far beyond the original reuse of similar cases to exploiting the collective experiences to tailor CBR with additional knowledge, to enable self-adaptation of the case knowledge, to offer flexible, opportunistic problem-solving through Agile CBR. From localised individual expertise to collective endorsement of problem-solving from a variety of experiences.

## Acknowledgments

I wish to thank Nirmalie Wiratunga, Jacek Jarmulak and Stewart Massie for contributions to this research. Ray Rowe, now at the Institute of Pharmaceutical Innovation, offered us the opportunity to work with the fascinating tablet formulation task that started this CBR research. Thanks are also given to many CBR researchers and anonymous reviewers who have helped to shape my research through comment and constructive feedback. This work has been supported in part by EPSRC (GR/L98015).

## References

1. Aamodt, A., Plaza, E.: Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AICOM* 7(1), 39–59 (1994)
2. López de Mántaras, R., et al.: Retrieval, reuse, revision, and retention in case-based reasoning. *Knowledge Engineering Review* 20(3), 215–240 (2005)

3. Richter, M.M., Aamodt, A.: Case-based reasoning foundations. *Knowledge Engineering Review* 20(3), 203–207 (2005)
4. Jarmulak, J., Craw, S., Rowe, R.: Self-optimising CBR retrieval. In: *Proc. 12th IEEE Int. Conf. on Tools with AI*, pp. 376–383. IEEE Press, Los Alamitos (2000)
5. Craw, S., Jarmulak, J., Rowe, R.: Maintaining retrieval knowledge in a case-based reasoning system. *Computational Intelligence* 17(2), 346–363 (2001)
6. Jarmulak, J., Craw, S., Rowe, R.: Using case-base data to learn adaptation knowledge for design. In: *Proc. 17th IJCAI*, pp. 1011–1016. Morgan Kaufmann, San Francisco (2001)
7. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence* 170(16–17), 1175–1192 (2006)
8. Kendall-Morwick, J., Leake, D.: Four heads are better than one: Combining suggestions for case adaptation. In: McGinty, L., Wilson, D.C. (eds.) *ICCBR 2009*. LNCS (LNAI), vol. 5650, pp. 165–179. Springer, Heidelberg (2009)
9. Wiratunga, N., Koychev, I., Massie, S.: Feature selection and generalisation for text retrieval. In: Funk, P., González Calero, P.A. (eds.) *ECCBR 2004*. LNCS (LNAI), vol. 3155, pp. 423–437. Springer, Heidelberg (2004)
10. Wiratunga, N., Lothian, R., Chakraborti, S., Koychev, I.: A propositional approach to textual case indexing. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) *PKDD 2005*. LNCS, vol. 3721, pp. 380–391. Springer, Heidelberg (2005)
11. Asimwe, S., Craw, S., Wiratunga, N., Taylor, B.: Automatically acquiring structured case representations: The SMART way. In: *Proc. 27th BCS Int. Conf. on Innovative Techniques and Applications of AI*, pp. 45–58. Springer, Heidelberg (2007)
12. Smyth, B., McKenna, E.: Modelling the competence of case-bases. In: Smyth, B., Cunningham, P. (eds.) *EWCBR 1998*. LNCS (LNAI), vol. 1488, pp. 208–220. Springer, Heidelberg (1998)
13. Massie, S., Craw, S., Wiratunga, N.: Complexity-guided case discovery for case based reasoning. In: *Proc. 20th AAAI*, pp. 216–221. AAAI Press, Menlo Park (2005)
14. Smyth, B., McKenna, E.: Competence models and the maintenance problem. *Computational Intelligence* 17(2), 235–249 (2001)
15. Craw, S., Massie, S., Wiratunga, N.: Informed case base maintenance: A complexity profiling approach. In: *Proc. 22nd AAAI*, pp. 1618–1621. AAAI Press, Menlo Park (2007)
16. Craw, S.: Agile case-based reasoning: A grand challenge towards opportunistic reasoning from experiences. In: *Proc. IJCAI 2009 Workshop on Grand Challenges in Reasoning from Experiences* (in press, 2009)
17. Goel, A., Ali, K., Donnellan, M., Gomez, A., Callantine, T.: Multistrategy adaptive navigational path planning. *IEEE Expert* 9(6), 57–65 (1994)
18. Nuxoll, A.M., Laird, J.E.: Extending cognitive architecture with episodic memory. In: *Proc. 22nd AAAI*, pp. 1560–1565. AAAI Press, Menlo Park (2007)
19. Woon, F.L., Knight, B., Petridis, M., Patel, M.: CBE-Conveyer: A case-based reasoning system to assist engineers in designing conveyor systems. In: Muñoz-Ávila, H., Ricci, F. (eds.) *ICCBR 2005*. LNCS (LNAI), vol. 3620, pp. 640–651. Springer, Heidelberg (2005)
20. Murdock, J.W., Goel, A.: Meta-case-based reasoning: Self-improvement through self-understanding. *Journal of Experimental and Theoretical Artificial Intelligence* 20(1), 1–36 (2008)
21. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Weber, R.O., Richter, M.M. (eds.) *ICCBR 2007*. LNCS (LNAI), vol. 4626, pp. 224–238. Springer, Heidelberg (2007)

# Black Swans, Gray Cygnets and Other Rare Birds

Edwina L. Rissland

Department of Computer Science  
140 Governors Drive  
University of Massachusetts  
Amherst, MA 01003-9264  
rissland@cs.umass.edu

**Abstract.** Surprising, exceptional cases — so-called black swans — can provoke extraordinary change in the way we do things or conceptualize the world. While it is not unreasonable to be surprised by a black swan, to be surprised by subsequent cases that are similar enough that they might cause the same sort of upheaval is unforgivable. The problem is how to reason about these almost novel, not totally unforeseen, subsequent cases that I call **gray cygnets**.

## 1 Introduction

From time to time there are unanticipated, exceptional cases that provoke extraordinary change in the way we do things or conceptualize the world. It is popular today to call such events *black swans* (e.g., [24]). They can be either positive or negative in the sense that they can expand or contract a conceptualization or approach. Not only do black swans trigger great changes, but also they are usually outside our conceptualization of the world and thus, come to us unheralded and seemingly out of the blue. They are said to exist in the “fat tail” of concepts.

While it is not unreasonable to be surprised by a black swan, to be surprised by subsequent cases that are similar to the black swan — in fact, similar enough that they might cause the same sort of upheaval — is unforgivable. As is often said: fool me once, shame on you; fool me twice, shame on me. If an entity — person or machine — cannot learn not to be fooled by essentially the same circumstance a second, or a third time, it is hardly intelligent.

The problem is how to spot and reason about cases that are similar enough to the original black swan that their effect might be similarly significant and exceptional. These almost novel, not totally unforeseen, subsequent cases are what I call **gray cygnets**.

## 2 The Gray Cygnet Problem: Four Research Issues

There are several aspects of the gray cygnet problem that are particularly relevant to case-based reasoning (CBR) and for which CBR is particularly well-suited. These include:

1. **Similarity assessment.** How similar to a black swan is similar enough for an event to be considered a gray cygnet? This is the classic CBR concern of “similarity assessment”. Further, given a new problem situation, how should a highly similar past black swan case be weighed against a highly similar past run-of-the-mill case? Which case, if any, should guide the system’s response? This is the classic “best case selection” issue.
2. **Hypotheticals.** How can hypotheticals be used to anticipate gray cygnets and explore the space of possible future cases and/or boundaries of current models, assumptions, policies, conceptualizations, etc. For instance, hypos based on a black swan can be created by simplifying it, pushing it to extremes, softening it to be a near hit or near miss, etc. and these can be used to plumb the instance space, particularly the fat tail where real data is sparse.
3. **Re-representation.** *How — and when — should the representation of a concept, plan, model, policy, etc. be changed in response to a black swan and gray cygnets?* For instance, how often should an underlying causal model be revised? What are the trade-offs between aggressive re-modeling with its risks of unstable model oscillations versus a “lazy” approach with its risks of using a model no longer valid? Should notions of what is typical or prototypical be revised? Should the validity of past analyses and solutions be revisited or even called into question?
4. **Analogical Mapping and Explanation.** *How should lessons learned from the black swan be mapped over or applied to a gray cygnet?* For instance, should one reason analogically. Should a new — “other” or anomaly — class that includes cygnet and swan be created, perhaps with minimal deep reasoning. What is the role of deeper explanation, and how should the exceptional treatment of a black swan and/or gray cygnets be explained?

Understanding the gray cygnet problem is important for several reasons. First, it is necessary for building intelligent systems that are not only robust in ever changing and challenging contexts but that can also be pro-active in such domains. Given that change is ubiquitous, it is necessary to understand how systems, including CBR systems, can learn in response to surprising, exceptional, and provocative events. Second, understanding the gray cygnet problem can shed insights on what is meant by concepts and how to represent them. Most concepts in the world are “messy” in the sense that they cannot be represented by universally valid necessary and sufficient conditions — compared with mathematical concepts, say — but yet there is often a great deal of regularity in them. This is the view pioneered by Rosch and colleagues in psychology research on notions of “typicality”, “prototypes”, and “atypical” instances (e.g., [22], [21]). How do swans and cygnets participate in such conceptualizations?

Similar viewpoints are found in legal philosophy concerning the differences between reasoning, and the nature of cases themselves, in the so-called “core of settled meaning”, where there are regularities and general heuristic truths and in the “penumbra”, where the cases are *sui generis* in the sense that each is somewhat unique and does not fit in the mold of cases in the settled core [5]. In CBR, an

analogous concern is the use of hybrid representations and reasoning, like rules or models and cases [20], [2], [7], [13], and more generally, mixed model-plus-extensional approaches for representation, such as frames or logic plus actual exemplars, which has long been of interest to me [18].

### 3 Examples of Black Swans and Gray Cygnets

Appellate law offers many examples of the scenario in which a black swan case creates an initial exception under which a line of gray cygnets emerge, so many so that eventually the accepted rule or approach becomes obsolete. For instance, the contract case *Thomas and Wife v. Winchester* (discussed in [12]) was the black swan that opened the way for a whole bevy of gray cygnets that eventually caused reversal of the “privity of contract” doctrine (that required a direct relationship between the manufacturer of a product and a party harmed by the product) and heralded legal changes now reflected in modern consumer rights law (that allows a consumer to recover damages for harm caused by a product, such as a car, bought through an intermediary like a dealer or store).

Another legal example is the 1925 prohibition era case of *Carroll v. United States*, 267 U.S. 132 (1925). This black swan created the so-called “automobile exception” under which police do not need a warrant to search a car. This exception flew in the face of the longstanding existing Fourth Amendment jurisprudence that said a warrant was necessary. The *Carroll* case created an exception that grew large enough to drive a truck through, and created a bifurcation in the concept of constitutionally valid searches — essentially the dichotomy of homes versus cars — that persisted for nearly 50 years, when the (not unimaginable) case of a house on wheels (i.e., a truck with a camper compartment) presented itself [17], [16]. Imagine if a hypo of a car used in a homelike way — say, for camping — had been raised in 1925. This is not an unimaginable hypo: think of Conestoga wagons, an example of a vehicle used for both transportation and homelike activities. Would the automobile exception have occurred or been justified as it was?

Black swans can occur in many disciplines, for instance, crashes or burst bubbles in economic markets. Financial black swans are the main focus of the Taleb book. The Great Depression is often treated as a black swan since it created great upheaval and was deemed a rare event. It is yet to be seen whether the recent “crash” in October 2008 will earn the title; it certainly created upheaval and change, but whether it was a true surprise is an open question.

Black swans can occur in medicine, often with quite negative results. For instance, the so-called Libby Zion case precipitated a drastic change in the way patients are treated in the Emergency Room and residents are trained and practice medicine. Libby Zion, an 18 year old college student, was seen in the ER for fever and mysterious jerking movements, admitted and sedated with an opiate (meperidine), and subsequently physically restrained and given further sedation (haloperidol) when she became agitated again. The next morning she died from cardiac arrest triggered by an alarming (107°) fever. While the cause of death was not determined definitively, it is suspected to have been



from drug interactions with the anti-depressant (phenelzine) she was taking. The only doctors that saw her were residents in training and they were routinely exhausted from long shifts. Training and protocols are now markedly different. ER physicians pale at the thought of creating a Libby Zion cygnet.

Even mathematics and science are not immune to black swans. Lakatos [10] discusses various treatments of some surprising counter-examples (e.g., Kepler's stellated dodecahedron) in the history of mathematics concerning Euler's formula. Weintraub [25] discusses examples from the long history of what heavenly objects are considered planets. For example, in the days of the early Greeks, Earth was not classified as a planet but the Sun was; the Copernican revolution switched their statuses. Pluto, discovered in 1930, was originally classified as a planet. Since then however, the discovery of a plethora of Trans-Neptune Objects has played havoc with the planet concept. Many astronomers feel that there are now far too many Pluto-like objects for them all to be called "planets"; so, to handle them, a new category called Plutinos was invented. If consistency is to be maintained, Plutinos along with Pluto should all be in or out of the concept class of planet. So the choice is either Pluto flips to being a non-planet or there are scores of new planets. Many feel that this enlarges the concept class too far.

In these examples, a drastic change is first heralded by a surprising, exceptional case (the black swan) that is then followed by a whole flotilla of subsequent, similar, exceptional cases (the gray cygnets). If one thinks of the set of positive instances of a concept, model, or a rule application as a disc, the black swan creates a puncture in it and the gray cygnets enlarge the hole. Eventually, there might not be much left of it. There might even be exceptions to the exceptions: positive islands within negative holes. Black swans can also occur near the boundary region and cause the concept to stretch or contract, thus flipping the classification of some nearby, borderline instances.

Gray cygnets, while perhaps not as "black" as the original event or case in the sense that they are obviously no longer as novel, can still be exceptional and have far-reaching ramifications, for instance by establishing a line of cases that solidify concept change. In precedent-based domains like law, gray cygnets tend to cite back to the original black swan case as justification, and once an exception has been allowed (e.g., not dismissed as a wrongfully-decided mistake), it is inevitable that more cases will be treated similarly. Eventually, this flock of exceptions might become the norm, when, as is often said, 'the exception swallows the rule'. In planning tasks, cygnets can indicate problems in underlying assumptions (e.g., unanticipated co-occurrences of circumstances).

## 4 Recognition of Swans and Vigilant Monitoring for Cygnets

There are two separate problems involving swans and cygnets: (1) recognizing a black swan, and (2) vigilant monitoring for subsequent, similar gray cygnets.

With regard to recognition, I note that it is often difficult to recognize a black swan event contemporaneously with its occurrence, especially in weak theory

domains, the forte of CBR, where one might need to await future commentary since it is often only in retrospect that there is recognition that one has occurred. This might have been the scenario in Levi example; the *Winchester* case was probably not considered a black swan when it was decided. It is often only when subsequent cases cite back to the black swan case as the enabling precedent — often with the rubric “landmark” case or use of its name as a shorthand to stand for a new way of thinking — that a black swan is truly recognized as such. In other domains, like planning, recognition might be more timely because the suspected black swan causes an immediately obvious surprise like an unanticipated failure or success [8]. That a case badly fits current expectations can lead to the hypothesis that a black swan event has occurred or is occurring [9]. However, it is typically only through closer analysis that one understands why it is exceptional, and this often requires more knowledge than just the case itself.

Once a surprising potentially paradigm-threatening case has occurred — even if not dubbed a black swan — one should not be snookered by similar, potentially disruptive, subsequent cases, that is, gray cygnets. One needs to be on the alert for them and invest in the reasoning required to understand their possible ramifications and monitor the case-stream for similar occurrences. While in some tasks, there is a rich enough domain knowledge to support pre-emptive measures, in many there is not, and watchful vigilance is the only alternative. That some cases turned out to be much ado about nothing and that vigilance wasted some resources does not, for me, obviate this approach since the cost-benefit trade-offs in many domains place a high penalty on missing cygnets. The worst possible approach in my opinion would be to dismiss the original black swan or subsequent cygnets as mere outliers or mistakes. This would open one up to making the same mistake twice, the hallmark of the opposite of an intelligent system.

## 5 Hypotheticals and Synthetic Cygnets

Creating hypotheticals is a way to approach the gray cygnet problem proactively. Hypos help perform a knowledge-based search of the case space (cf, statistical sampling) and provide useful ersatz data when real data is sparse. They can be used to probe the current conceptual regime. A hypothetical gray cygnet can be particularly telling.

One way to create synthetic cygnets is to take a black swan as a starting “seed” case and modify it heuristically in ways that are important in the domain to create derivative hypos or an entire “constellation” of them. Methods based on “dimensions” have been used to implement this approach in HYPO-style systems as well as analyze how experts do it [19], [17]:

- (i) Make a case weaker or stronger along a dimension
- (ii) Make a case extreme
- (iii) Enable a near miss (near negative)
- (iv) Dis-able a near win (near positive)

- (v) Add a closely coupled aspect
- (vi) Add potentially conflicting aspects
- (vii) Weaken a clear win (solid positive) to be a near win.

For instance, in the warrantless search area, major dimensions are “expectation of privacy” and “exigency”. One can strengthen a fact situation on the first dimension by making a vehicle more homelike by adding features that indicate use of the vehicle’s interior for living (e.g., privacy curtains, bedding, cooking equipment). One can weaken the exigency dimension by compromising a vehicle’s mobility (e.g., by having it hooked up to utilities in an RV park, parked up on blocks). One can create a conflict hypo by making a situation extremely strong on both, for instance, a fully equipped RV capable of traveling at highway speeds. Such hypos were actually used in actual Supreme Court oral arguments by the Justices to explore the ramifications of potential analyses and decisions [17].

Besides dimension-based techniques, there are others used in the CBR community, for instance those based on distance metrics. For instance, using the k-NN metric, one can create a mid-point hypo or the “nearest unlike neighbor” or “NUN” [15], [14].

## 6 Responsive Re-representation

Once a black swan and gray cygnets have been discovered, the issue is how to revise the conceptual regime. Some responses can mimic those discussed by Lakatos, for instance, “surrender” (reject the old regime or model entirely) or “monster barring” (bar them from being a legitimate case in the instance space).

In Anglo-American common law, the standard response is to add the black swan to the case base and then simply allow exceptional gray cygnet cases to accrete in the case base so long as they are justified in a precedent-based manner (e.g., citing the black swan). The law does not explicitly remove from the case base cases that are discredited by the black swan, or that rely on them. They are still in the case base, but it is up to legal practitioners to know that they might no longer be good law and of little precedential value. Eventually, the set of black swan plus gray cygnets can dominate the case-base or be declared by some (appellate) court to be the correct approach. Levi’s privity of contract example and the automobile exception examples illustrate these phenomena.

In statutory law where there are rules as well as cases, the black swan and its cygnets can create exceptions, temporary updates, or implicit work-arounds to the statutory rule. Eventually the body that wrote that statute (e.g., Congress) can re-write the rule to reflect the swan and cygnets in a revised statute. In some cases, like tax law, new regulations and advisories can be issued to say how such cases are to be handled. My lab’s hybrid rule and case-based system CABARET used cases in this way (see [20], [23]).

In a domain with an explicit theory, an appropriate theory revision mechanism can be used. For instance, if the domain theory is an ontology, it can be revised by splitting or removing subclasses. If there is a causal model, causal pathways

can be added or removed, parameter ranges restricted or expanded, new variables introduced, etc. Of course, such revisions can be difficult because they can involve age-old AI problems like credit assignment.

In my lab, we continue to investigate *hybrid CBR and representations for concepts, domain models, policies, etc.* We use (a) a general summary representation **PLUS** (b) a set of exemplars, called the **E-set**, a selective subset of the case knowledge base. The summary representation might take the form of definitional rules, HYPO-style dimensions, prototypes, or causal models. The E-set includes both positive and negative instances and is the repository for “atypical” or “penumbral” or “surprising” cases. Black swans and gray cygnets can be captured and stored as members of the E-set. Re-representation can occur through accretion of cases in the E-set and/or explicit revision of the summary aspect. Management of the E-set is an interesting problem in itself. For instance, when a summary representation is revised, should the old E-set be retained for future use: totally, in part, not at all?

Finally, mapping over the lessons learned from a black swan or gray cygnet involves analogy and explanation. In addition to HYPO-style analogical reasoning [1], there is much relevant work by Gentner and Forbus on “structure mapping” that can provide an approach to analogy [4],[3]. It was fundamental to Branting’s GREBE project in a statutory legal domain [2]. A different and also highly effective approach was used in Hammond’s CHEF [6]. Of course, there is much powerful research by Roger Schank, David Leake and others on explanation involving cases [11].

## References

1. Ashley, K.D.: Modeling Legal Argument: Reasoning with Cases and Hypotheticals. MIT Press, Cambridge (1990)
2. Branting, L.K.: Building Explanations from Rules and Structured Cases. International Journal of Man-Machine Studies (IJMMS) 34(6), 797–837 (1991)
3. Falkenheimer, B., Forbus, K.D., Gentner, D.: The Structure-Mapping Engine: Algorithm and Examples. Artificial Intelligence 41(1), 1–63 (1989)
4. Gentner, D.: Structure-Mapping: A Theoretical Framework for Analogy. Cognitive Science 7(2), 155–170 (1983)
5. Hart, H.L.A.: Positivism and the Separation of Law and Morals. Harvard Law Review 71, 593–629 (1958); reprinted in Hart, Essays in Jurisprudence and Philosophy. Oxford (1983)
6. Hammond, K.J.: Case-Based Planning: Viewing Planning as a Memory Task. Academic Press, London (1989)
7. Hastings, J., Branting, L.K., Lockwood, J.: CARMA: A Case-Based Rangeland Management Advisor. AI Magazine 23(2), 49–62 (2002)
8. Horling, B., Benyo, B., Lesser, V.: Using Self-Diagnosis to Adapt Organizational Structures. In: Proc. AGENTS 2001, Montreal (2001)
9. Horvitz, E., Apacible, J., Sarin, R., Liao, L.: Prediction, Expectation, and Surprise: Methods, Designs, and Study of a Deployed Traffic Forecasting System. In: Proc. UAI (2005)
10. Lakatos, I.: Proofs and Refutations. Cambridge (1976)

11. Leake, D.B.: *Evaluating Explanations: A Content Theory*. Lawrence Erlbaum, Mahwah (1992)
12. Levi, E.H.: *An Introduction to Legal Reasoning*. Chicago (1949)
13. Marling, C., Rissland, E.L., Aamodt, A.: Integrations with case-based reasoning. *Knowledge Engineering Review* 20(3), 241–246 (2005)
14. Massie, S., Wiratunga, N., Craw, S., Donati, A., Vicari, E.: From anomaly reports to cases. In: Weber, R.O., Richter, M.M. (eds.) *ICCBR 2007*. LNCS, vol. 4626, pp. 359–373. Springer, Heidelberg (2007)
15. Massie, S., Craw, S., Wiratunga, N.: Complexity-Guided Case Discovery for Case Based Reasoning. In: *Proc. AAAI 2005*, pp. 216–221 (2005)
16. Rissland, E.L.: AI and Similarity. *IEEE Information Systems* 21(3), 39–49 (2006)
17. Rissland, E.L.: Dimension-based Analysis of Hypotheticals from Supreme Court Oral Argument. In: *Proc. Second International Conf. on AI and Law (ICAIL 1989)*, Vancouver, BC, pp. 111–120 (1989)
18. Rissland, E.L.: *Epistemology, Representation, Understanding and Interactive Exploration of Mathematical Theories*. Ph.D. dissertation, Department of Mathematics, MIT (1977)
19. Rissland, E.L., Ashley, K.D.: Hypotheticals as Heuristic Device. In: *Proceedings Fifth International Conference on Artificial Intelligence (AAAI 1986)*, pp. 289–297 (1986)
20. Rissland, E.L., Skalak, D.B.: CABARET: Statutory Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies (IJMMS)* 34, 839–887 (1991)
21. Rosch, E., et al.: Basic Objects in Natural Categories. *Cognitive Psychology* 8(3), 382–439 (1976)
22. Rosch, E., Mervais, C.B.: Family Resemblance: Studies in the Internal Structure of Categories. *Cognitive Psychology* 7, 573–605 (1975)
23. Skalak, D.B., Rissland, E.L.: Arguments and Cases: An Inevitable Intertwining. *Artificial Intelligence and Law* 1(1), 3–44 (1992)
24. Taleb, N.N.: *The Black Swan*. Random House (2007)
25. Weintraub, D.: *Is Pluto a Planet?* Princeton (2007)

# Case Retrieval Reuse Net (CR2N): An Architecture for Reuse of Textual Solutions

Ibrahim Adeyanju<sup>1</sup>, Nirmalie Wiratunga<sup>1</sup>, Robert Lothian<sup>1</sup>,  
Somayajulu Sripada<sup>2</sup>, and Luc Lamontagne<sup>3</sup>

<sup>1</sup> School of Computing  
Robert Gordon University, Aberdeen, Scotland, UK  
{iaa,nw,rml}@comp.rgu.ac.uk

<sup>2</sup> Department of Computing Science,  
University of Aberdeen, Aberdeen, Scotland, UK  
yajj.sripada@abdn.ac.uk

<sup>3</sup> Department of Computer Science and Software Engineering,  
Université Laval, Québec (Québec), Canada  
luc.lamontagne@ift.ulaval.ca

**Abstract.** This paper proposes textual reuse as the identification of reusable textual constructs in a retrieved solution text. This is done by annotating a solution text so that reusable sections are identifiable from those that need revision. We present a novel and generic architecture, Case Retrieval Reuse Net (CR2N), that can be used to generate these annotations to denote text content as reusable or not. Obtaining evidence for and against reuse is crucial for annotation accuracy, therefore a comparative evaluation of different evidence gathering techniques is presented. Evaluation on two domains of weather forecast revision and health & safety incident reporting shows significantly better accuracy over a retrieve-only system and a comparable reuse technique. This also provides useful insight into the text revision stage.

## 1 Introduction

Textual Case Based Reasoning (TCBR) solves new problems by reusing previous similar problem-solving experiences documented as text. TCBR is a subfield of Case Based Reasoning (CBR) but has evolved as a specialized research area due to challenges associated with reasoning with textual attributes as opposed to structured attributes consisting of numeric and symbolic values [1].

In structured CBR, a case is typically described using a fixed number of attributes; therefore, the reuse stage will propose a solution containing values for these fixed attributes. Although a solution is also proposed for reuse in TCBR, number of attributes differ when the solution is textual and its decomposition into sections (keywords, phrases or sentences) is viewed as attributes. The number of sections in a retrieved textual solution is also likely to differ from the actual solution. Therefore, the reuse stage for TCBR must identify sections of

a solution text that are relevant (reusable) to a given problem. The rest are candidates for revision which may take the form of deletion.

In this paper, we present a novel architecture for textual reuse which identifies sections of a retrieved text as reusable or alternatively needing revision. Our architecture extends the Case Retrieval Net (CRN) and establishes evidence in support of either reuse or revision by analysing the retrieval neighbourhoods. We design an algorithm to formalise our architecture and evaluate it on two application domains: post-editing of weather forecast texts and health and safety incident reporting. Common to both domains is that the problem and solution are in textual form. However, the domains also exhibit different textual characteristics such as in vocabulary size, problem and solution vocabulary overlap and the use of synonyms. Such differences allows us to evaluate the transferability of our technique across domains.

Section 2 discusses related work in CBR reuse and distinguishes textual from structured reuse. We then explain details of our novel architecture for textual reuse in Section 3 and compare it with an existing technique in Section 4. This is followed by experimental setup and discussion of results in Section 5 with conclusions in Section 6.

## 2 Related Work

The concept of CBR reuse was introduced in [2] to cover different ways in which knowledge is processed from retrieved cases prior to revision. In broad terms, this consists of generative and transformational reuse. Generative reuse (also called replay) involves a trace of the retrieved solution in the context of the new problem. A search-based approach to generative reuse was proposed for configuration tasks in [3]. Solutions in this domain consist of a complex structure of elements which are captured as *states*; a domain-specific representation of a partially specified solution. A solution to a given problem is therefore generated by searching the space of solutions guided by the retrieved solution. This technique is not directly applicable to textual content because of the difficulty in capturing a partially specified text content without losing its contextual meaning within the entire solution.

Transformational reuse on the other hand is one in which the contents (all attributes and values) of a retrieved solution are copied verbatim or aggregated by consensus of retrieved solutions. This technique was exploited for automatic story plot generation [4]. A plot structure is obtained by reusing stories from a case base of tales and an ontology of explicitly declared relevant knowledge. The ontology enables measuring of semantic distance between words/structures in the query and previous problems while the solution in each case consist of fairy tale texts analysed and annotated according to Propp's morphology. Natural Language Generation (NLG) techniques are then used to describe the story plot in natural language. Although the story generated is a complete sketch of the plot, it assists screen writers in fast prototyping of story plots which can easily be developed into a story. The approach is knowledge intensive and use of a

domain specific ontology limits its transferability. However, it paves the way to exploit other interesting synergies between NLG and CBR.

A restricted form of textual reuse is presented for report writing applied to the air travel incident domain [5]. Here, textual cases consist of incident reports with one or more paragraphs grouped under a specific heading as a section. The most similar document to a query is retrieved and textual reuse is facilitated for each section of the retrieved report. This is done by presenting a cluster of other documents containing similar text under the same heading. This technique ignores the context of each section within the entire report which leads to unuseful clusters. The approach is restrictive since it cannot be used in the absence of common section headings across the set of documents.

The drawbacks observed in the work reviewed above are addressed by a text reuse technique called *Case Grouping (CG)* [6]. The technique demonstrated on a semi-automated email response application involves reuse of previous email messages to synthesize new responses to incoming requests. A response is a sequence of statements satisfying the content of a given request and requires some personalization and adjustment of specific information to be reused in a new context. The reuse technique annotates sentences as reuse if there is sufficient evidence that similar past problems contain this sentence. The evidence is quantified by dividing the case base into two clusters that contain similar sentence and those that don't. Query similarity to a centroid case formed for each cluster determines whether or not to reuse. The centroid case has the average value for each feature across all cases in a cluster. The use of similarity knowledge to guide reuse/revision is novel; however, use of centroids to achieve this is less desirable because two clusters could have the same centroid if the spread of cases result in similar intra-cluster distance ratios. Also, use of the entire casebase to form clusters implies that the computation is influenced by cases which have no similarity to the query nor to the retrieved case.

### 3 Case Retrieval Reuse Net (CR2N)

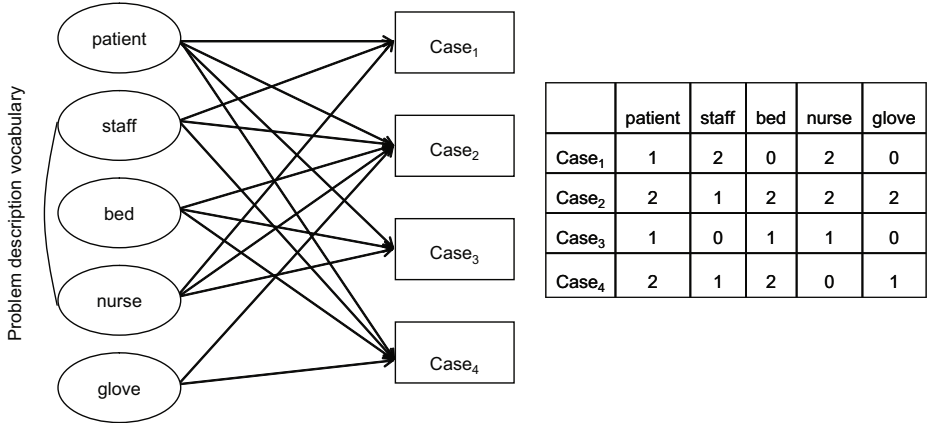
Our approach to reuse involves automated annotation of retrieved solution text as relevant or not. Essentially, textual units (keywords, phrases, sentences etc) annotated as relevant suggests that they can be reused without revision. In order to achieve this, we propose an extension to the CRN architecture called CR2N. The CR2N architecture consists of two CRNs: the original Case Retrieval Net (CRN) [7] which indexes the problem vocabulary and a second CRN referred to as Case Reuse Net (CReuseNet) which indexes the solution vocabulary.

#### 3.1 Case Retrieval Net (CRN)

A CRN is a memory model that can efficiently retrieve a relatively small number of relevant cases from a case base. The model in its basic form was proposed by Lenz & Burkhard [7] although several extensions to the basic CRN such as the lazy propagation CRN [8], Microfeature CRN [8] and Fast CRN [9] have been



proposed. The CRN is efficient because it avoids exhaustive memory search and can handle partially specified queries; complete because it assures that every similar case in memory is found during retrieval[7]. It is also flexible as there are no inherent restrictions concerning the circumstances under which a particular piece of knowledge can be recalled and this is particularly useful for text.



**Fig. 1.** Partial CRN for the health & safety dataset with matrix representation

The CRN uses a net-like case memory to spread activation for retrieval of similar cases to a query. It consists of four components: case nodes, Information Entities nodes (IEs), relevance arcs and similarity arcs. An IE consists of an attribute-value pair and a case consists of a set of IEs. A relevance arc connects IEs to cases and shows the presence and strength of an IE in a case while a similarity arc connects two IEs and indicates how similar an IE is to another. A case retrieval is performed by activating IE nodes which occur in a given query, propagating this activation according to similarity through the net of IEs and aggregating activation in the associated case nodes. Cases are ranked according to this aggregation and solution from the top k cases are retrieved.

When used in TCBR, each IE node is used to represent a single textual unit (keyword, phrase or sentence) depending on the granularity of indexing and similarity matching. Similarities between the textual units are then captured by the similarity arcs. A trivial CRN built for our incident reporting application is illustrated in figure 1 with its corresponding matrix representation. The figure shows how health & safety keywords relate to incident cases. A relevance arc connects an IE to a case when the keyword associated with the IE is contained in the case. For example the keywords “patient”, “staff”, “bed”, and “glove” occur in case Case<sub>4</sub>. The weight on the arc typically denotes the importance of the keyword in a case. Here, we use term frequency weighting and each row in the matrix relates to a case represented as a feature vector. The similarity arc between “staff” and “nurse” indicates that the two keywords are similar

and could be learnt using word co-occurrences or from an ontology. Aggregation network activations are implemented using matrix multiplication [9].

### 3.2 From CRN to CR2N

The trivial example used to illustrate the components of the CR2N in figure 2 has a case base of six cases and five/four keywords in the problem/solution vocabulary respectively. The CRN retrieves the most similar case(s) to a query while the Case Reuse Net (CReuseNet) generates text annotation on the proposed solution. CRN represents the problem vocabulary of indexed cases as a mapping between IE nodes and cases containing such IEs. Case nodes are denoted as  $C$  and the problem description IEs are denoted as PIE. Mapping of IEs onto cases are shown as relevant arcs while the similarity arcs indicate the similarity between IEs. Solution description IEs in the CReuseNet are denoted as SIE to differentiate these from problem description IEs.

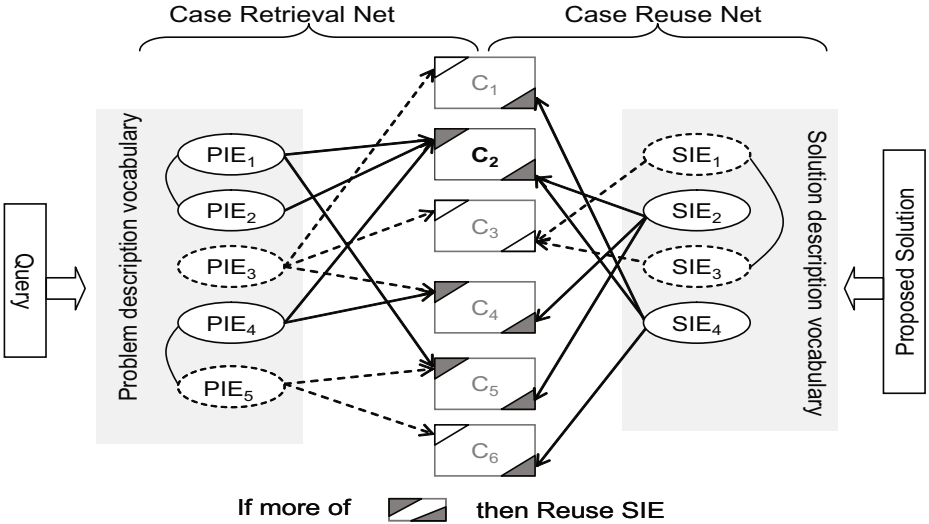


Fig. 2. The Case Retrieval Reuse Net (CR2N) architecture

A query spreads activation in the CRN through its PIEs. The most similar case is identified as that having the highest aggregation of activations ( $C_2$  in figure 2). Each SIE from the most similar case then spreads activation in the CReuseNet to determine its reusability to the query. We decide the reusability of the SIE by comparing two retrieval sets:  $RS_1$ , the set of cases activated in the CRN by a query and  $RS_2$ , the set of cases activated by an SIE in the CReuseNet. A large intersection between  $RS_1$  and  $RS_2$  implies reuse of SIE otherwise revise. In other words, an SIE is reusable if a majority of the cases it activates in CReuseNet have already been activated in the CRN. For example in figure 2,

$C_2$  (most similar to the query) contains  $SIE_2$  &  $SIE_4$ .  $SIE_2$  is determined to be reusable because all cases ( $C_2$ ,  $C_4$  &  $C_5$ ) activated by the query in the CRN are also activated by the  $SIE_2$  node. On the other hand,  $SIE_4$  is likely to need revision because it only activates one ( $C_2$ ) out of the three cases activated by the query in the CRN.

## 4 Evidence for Annotations: Neighbouring vs. All Cases

We illustrated a simple view of our reuse architecture in figure 2 with six cases being considered in the creation of the sets  $RS_1$  and  $RS_2$ . However, a casebase will contain many more cases. Since these sets provide evidence for our annotation judgements, we need to establish how these sets are formed for a larger casebase size. Clearly, it is sensible to use local neighbourhoods for evidence in our reuse computation rather than the entire casebase.

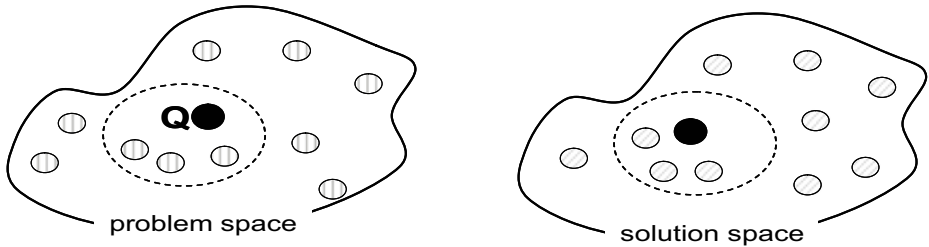


Fig. 3. Neighbourhoods in the problem and solution spaces

It is natural to use the problem space neighbourhood (i.e. query’s neighbours) since similar problems should have similar solutions. This implies that evidence for reuse is computed using the query’s neighbouring cases. For instance in figure 3,  $Q$  appearing in the problem space represents a query and the filled-in circle represents the best match case. The evidence for reuse/revise can be obtained from four cases nearest to the query as indicated by the outline around  $Q$  in the problem space. Alternatively, we could focus on the solution space neighbourhood consisting of the retrieved solution’s neighbours. The use of this neighbourhood allows each retrieved SIE to be put in context of the entire solution during reuse computation. Such contextualisation for example enables the solution keyword “plate” in “food plate” to be disambiguated from “plate kit” used in surgery when applied to our health and safety incident reports.

### 4.1 Text Reuse with Case Grouping

Case Grouping ( $CG$ ) [6] is a strategy which obtains evidence for textual reuse from the entire case base in its original form. An SIE in a proposed solution is annotated as reusable if there is sufficient evidence that similar past problems

also contained it in their solution. The key idea is the use of all cases in the casebase to gather evidence for or against reusing a proposed solution SIE. For each SIE in the retrieved solution, the case base is divided into 2 clusters: support and reject. The support cluster consists of cases that contain a similar SIE in their solution while the reject cluster contains cases that don't. A centroid case is then formed for each cluster by combining problem vectors of every case in the cluster. An SIE is annotated as reusable if the support centroid case is more similar to the query than the reject centroid; otherwise, the SIE is annotated as revise. Notice here that evidence for annotation is based on the whole casebase.

## 4.2 Text Reuse with CR2N

CR2N emphasizes the use of local neighbourhoods as opposed to CG's use a global view of the whole casebase. We formalise our CR2N architecture as an algorithm (see Figure 4) to automatically generate our textual reuse annotations (i.e. reuse/revise). The algorithm uses a generic CRN function to retrieve cases given a partial case description and an indexing vocabulary. There are two CRN function calls, with the first retrieving over the problem vocabulary,  $V_p$ , and the second over the solution vocabulary,  $V_s$ . The retrieval sets returned by the CRNs are qualified by two further Select functions: SelectK returns the top k cases, and SelectT returns all cases with similarity above a specified threshold. Although other retrieval mechanism (e.g. feature based matching) can be used, we employed CRN because of its efficiency on larger applications and its similarity arcs allows for more semantic retrieval.

The best match case  $C_{best}$ , is identified by retrieving over  $V_p$  in response to a query Q. Here Q is a case consisting of just the problem description and  $RS_1$  is the resultant retrieval set by retrieving over  $V_s$  with the retrieved solution from  $C_{best}$ . The reuse stage involves iterating over the proposed textual solution content (i.e.  $C_{best}$ 's solution) to identify and annotate relevant parts. Like the second CRN call, the third CRN retrieves cases over the solution vocabulary given some partial solution text, which is formally denoted as a set of solution IEs or  $\{sie_i\}$  in figure 4. The resultant retrieval set is  $RS_2$ . It should be noted that  $\{sie_i\}$  must be a subset of  $C_{best}$ 's solution.

A solution IE is reusable by the query if cases containing it are similar to the query. In other words we want to establish if cases with similar problem descriptions to the query also contain the solution IE of interest,  $\{sie_i\}$ . For this purpose the retrieval sets  $RS_1$  and  $RS_2$  are compared. The intersection of these sets contain cases (AS) that have similar solution to the retrieved solution and also contain the  $sie_i$ , whilst the set difference identifies cases (BS) that are similar to the retrieved solution but not containing  $\{sie_i\}$ . The annotation is conditioned on the average similarity of the query to cases in the intersection ( $\bar{S}_A$ ) versus that of the set differences ( $\bar{S}_B$ ). The solution is determined to be reusable if  $\bar{S}_A$  is greater than  $\bar{S}_B$  else it needs revision.

The  $SelectK(CRN(V_s, C_{best}), k)$  function retrieves k cases similar to the retrieved solution. The function thereby allows the retrieved solution's overall context to be taken into account even when IEs are used for activation one at a time.

---

$CB = \{C_1, \dots, C_n\}$ , set of cases in the case base  
 $V_p = \{pie_1, \dots, pie_m\}$ , set of problem IEs in CB  
 $V_s = \{sie_1, \dots, sie_l\}$ , set of solution IEs in CB  
 $C = \{P, S\}$ , where  $(C \in CB) \wedge (P \subset V_p) \wedge (S \subset V_s)$   
 $Q =$  a query, where  $Q \subset V_p$   
 $k =$  local neighbourhood used for reuse calculation, where  $k \leq n$

```

 $C_{best} = \text{SelectK}(\text{CRN}(V_p, Q), 1)$ 
 $RS_1 = \text{SelectK}(\text{CRN}(V_s, C_{best}), k)$ 
for each  $\{sie_i\} \in C_{best}$ 
   $RS_2 = \text{SelectT}(\text{CRN}(V_s, \{sie_i\}), \sigma)$ 
   $AS = RS_1 \cap RS_2$ 
   $BS = RS_1 \setminus RS_2$ 
   $\bar{S}_A = \frac{1}{|AS|} \sum_{a \in AS} \text{Sim}(a, Q)$ 
   $\bar{S}_B = \frac{1}{|BS|} \sum_{b \in BS} \text{Sim}(b, Q)$ 
  if  $\bar{S}_A > \bar{S}_B$ 
  then
    REUSE  $\{sie_i\}$  (relevant to the query)
  else
    REVISE  $\{sie_i\}$  (irrelevant to query)

```

---

**Fig. 4.** The CR2N Algorithm

Alternatively, neighbours of the query could have been used but our previous experiments reported in [10] showed that using neighbourhoods from solution space perform better than the problem space. The use of a specified  $k$ -neighbourhood increases the efficiency of the algorithm since a smaller number of cases are used for reuse computation. Small values of  $k$  ensure that a local neighbourhood is used for reuse computation and remove the influence of cases with little similarity to the retrieved. This is important since these cases could negatively affect the reuse computation because they reduce average similarity of AS.

The CR2N algorithm is generic because IEs can represent any form of textual units (keywords, phrases, sentences etc). Also the algorithm could still be used if each IE represents a keyword and we want to annotate larger textual units like sentences or paragraphs. This is done by using all keywords in the textual unit as a set for activation in the function  $\text{SelectT}(\text{CRN}(V_s, \{sie_i\}), \sigma)$ . The best values for parameters  $k$  and  $\sigma$  on a given domain must be established empirically.

### 4.3 Distinguishing CR2N from CG

CR2N is similar to CG (see section 4.1) in that both exploit the indirect relation between a query and each textual unit in a retrieved solution by forming two sets of cases (AS/support & BS/reject). However, CR2N addresses drawbacks identified in CG as follows.

1. CR2N uses knowledge from a specified local neighbourhood to determine reusability of a solution’s textual unit instead of an entire case base used in CG. This removes the influence of cases that are dissimilar to the retrieved case during reuse computation.
2. Average similarity of cases in each group to the query is employed for reuse/revise evidence in CR2N rather than centroid vectors used in CG. This is more intuitive since it takes into account similarity to the query of each case individually rather than as a group of cases.
3. Unlike CG, retrieval and reuse are integrated into a single architecture.

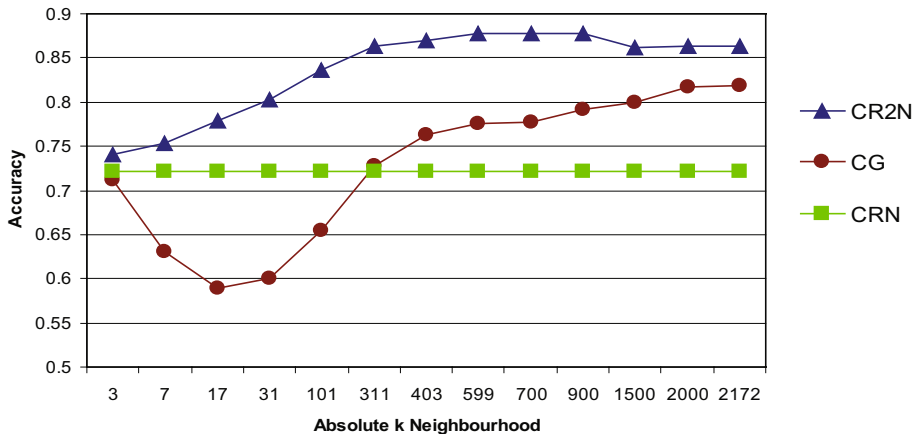
## 5 Evaluation Methodology

We evaluate the effectiveness of the reuse strategies by measuring accuracy of their annotations. We chose a retrieve-only system as our baseline since reuse succeeds retrieval and its use should improve upon retrieval results. We are also interested in the effect of different neighbourhood sizes ( $k$ ) on reuse performance, we therefore repeated our experiments for increasing values of  $k$ . We compared the baseline with two textual reuse algorithms.

1. CR2N as explained in section 4.2
2. CG, as reviewed in section 4.1 but modified to use neighbourhoods (instead of the entire casebase) of the query to make it comparable to CR2N

We use a ten-fold cross validation and cosine similarity computation at both retrieval and reuse stages. Each IE in the CR2N represents a keyword from our domain vocabulary. We chose keywords as our textual units to be annotated because the size of each retrieved solution text in our application domains is small (typically 1 sentence with an average of 7 keywords).

We evaluate effectiveness of the CR2N using average accuracy, precision and recall. Our underlying hypothesis is that an effective reuse of retrieved similar cases would enhance revision and should perform better than the retrieve-only baseline. Accuracy of the CR2N is measured as a ratio of retrieved keywords correctly annotated as reuse/revise to the total number of keywords retrieved. We measure precision as a ratio of the number of keywords from the actual solution present in the proposed solution to all keywords in proposed solution. Recall is a ratio of keywords from actual solution present in the proposed solution to all keywords in actual solution. These measures are commonly used to evaluate TCBR systems [11] but have practical limitations as they are surface measures devoid of most semantics in the context of a sentence. Accuracy shows predictive performance of the CR2N and the retrieval precision is used as baseline accuracy since all retrieved keywords are deemed reusable if no annotation is done. On the hand, precision/recall indicates overall performance of our TCBR system when keywords annotated as revise by CR2N are deleted. A higher reuse precision with comparable recall over a retrieve-only system would indicate better effectiveness for a simplified TCBR system in which only delete operations are carried out during revision. However, a complete revision stage will also include substitute and insert edit operations; we intend to tackle this in our future work.



**Fig. 5.** Accuracy results for CR2N, CG and CRN in weather forecast revision

The problem and solution texts are preprocessed using the GATE library, available as part of the jCOLIBRI [12] framework. These attributes are divided into keywords using the GATE Splitter. Suitable stop words are also removed and keywords stemmed to cater for morphological variations.

## 5.1 Weather Forecast Revision

The wind dataset was extracted from a post-edit corpus [13] of an NLG weather forecast system called Sumtime Mousam (SM). The dataset consists of weather forecast text generated from numerical data by SM and its edited form after revision by domain experts. A case in our experiments therefore consists of the NLG system generated text (Unedited Text) as problem and its revised form by domain experts (Edited text) as solution.

The SM weather corpus has the following peculiar properties:

- The problem text is more similar to its solution text in a single case than to any problem text from other cases. This means that the problem & solution vocabularies are identical unless forecasters introduce new terminology. Although this is unlike most TCBR applications where the problem & solution have very few vocabulary in common (e.g. incident report datasets [14][15]), we expect that similar edit operations are applicable on solution texts.
- The indexing vocabulary is small i.e. 71/ 140 keywords in problem/ solution vocabulary respectively.
- The problem (Unedited text) is very consistent because it is generated by an NLG system with abstracted rules but the solution is not as consistent and may contain typing errors (e.g. middnigh, lessbecoming).

A total of 2414 cases (from 14690) were extracted for experiments and we ensured that the average size of problem/solution text is about 1 sentence since the

reuse techniques were tested at keyword granularity. Figure 5 shows an accuracy graph comparing the retrieved similar solution (CRN), CR2N and CG from our experiments with the wind forecast dataset. The average accuracy of the CR2N clearly outperforms the baseline (precision of the retrieved solution) and CG as its curve is above them. Also, CR2N’s accuracy increases with  $k$  neighbourhood of the retrieved solution, attains its best value when  $k=700$  (about one-third of 2172 cases in the training set) and starts to decrease thereafter. This increase in accuracy with  $k$  can be attributed to the CR2N having more contextual knowledge to predict the reuse/revise of a keyword better. The decrease thereafter establishes the fact that comparison of local neighbourhoods is sufficient rather than the entire case base. The local neighbourhood is large because the vocabulary is small, therefore, majority of cases have common keywords in their solution text. CG shows a different trend; the accuracy is initially below that of the baseline (until  $k=17$ ) but increases subsequently outperforming the baseline (after  $k=311$ ). The initial decrease could be attributed to the misleading and unpredictable evidence from the use of centroids even when a smaller number of cases are used to create the clusters.

The average precision/recall values plotted against the absolute neighbourhood values is shown in figure 6. These curves show a similar pattern in effectiveness with the CR2N surpassing the others. The average recall of the CR2N becomes comparable to the average retrieval recall when  $k=900$  but with higher precision. The recall of CR2N cannot be greater than the retrieval recall as keywords annotated as revise are currently treated as deletes. The CR2N’s performance is generally above that of CG on the graph except when  $k=3$  and

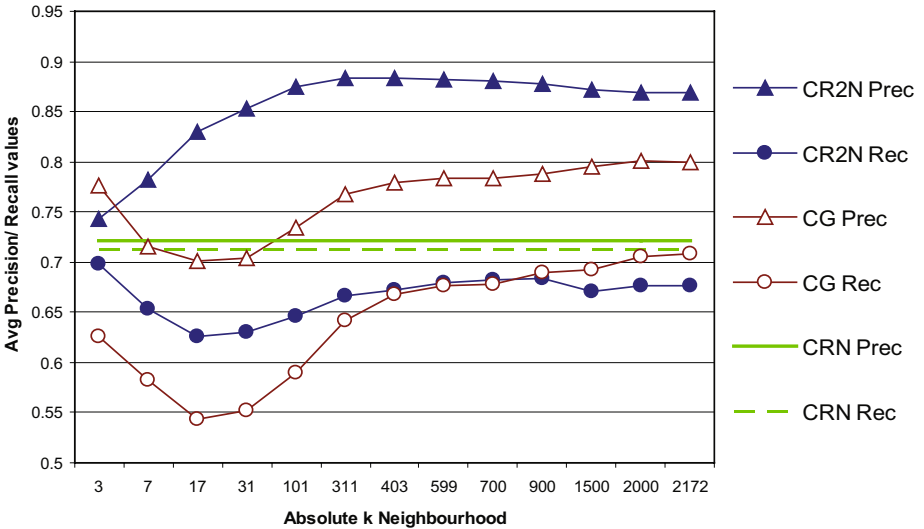


Fig. 6. Precision/Recall results for CR2N, CG and CRN in weather forecast revision



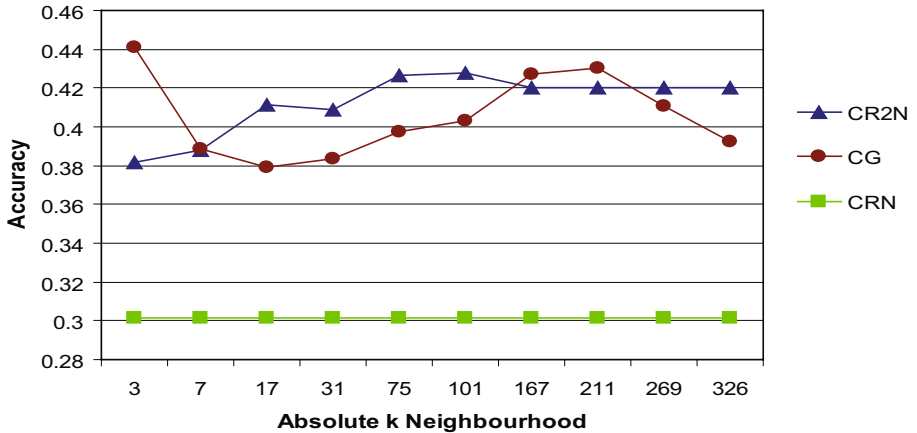


Fig. 7. Accuracy results for CR2N, CG and CRN in H&S incident reporting

$k > 700$  for average precision and recall respectively. The higher average recall values show that CG is more conservative because it tends to reuse most keywords in a retrieved solution.

## 5.2 Health and Safety Incident Reporting

We also evaluated our technique on health and safety (H&S) incident reports from hospitals provided (by NHS Grampian). A report consists of a textual description of the incident and the action taken by the health personnel on duty. Each record is also labelled with 1 of 17 care stage codes which identifies a group of records such as accidents that result in personal injuries, incidents during treatment or procedures etc. Our intention is to build a TCBR system that assists less experienced health personnels when resolving/recording incidents by using previous similar experiences. Therefore, the incident description serves as our problem while the solution is the action taken to resolve the incident for each case in our experiments.

Unlike the weather forecast revision domain, health and safety incident reporting is a typical TCBR application where problem and solution vocabulary share little in common and indexing vocabulary is large (e.g. 732 keywords in solution vocabulary). Also, both problem and solution texts may contain typing errors since they are manually recorded by humans. We extracted a total of 362 cases that were grouped under a similar care stage code and having just 1 sentence in both the problem and solution texts. This allows us not only to evaluate our reuse technique at keyword granularity but makes it comparable to results from the weather domain. During evaluation, synonym keywords were matched using WordNet [16] as well as keywords with the same lemma but different stems (e.g gave and given).

Figure 7 shows an average accuracy graph comparing the baseline (CRN), CR2N and CG from our experiments with the H&S incident reports. The performance of the reuse techniques exceed the baseline as shown by their accuracy

plots. There is no clear distinction between CR2N and CG’s performance but CR2N is marginally better with 6 wins out of the ten neighbourhood sizes evaluated. Overall, CR2N is most consistent with an initial increase in accuracy followed by a decrease that tapers as the neighbourhood size increases. This indicates an optimal performance when neighbourhoods are used for reuse computation as opposed to the entire case base. CG on the other hand shows a less predictable pattern with increasing neighbourhood size. In particular, the initial high accuracy is surprising. A closer look at this point (CG at  $k = 3$ ) shows that one of the two clusters used for centroid creation was always absent leading to random behaviour that was advantageous in this instance.

CR2N’s precision outperforms those of CG and CRN (see figure 8). However, the average recall of CG is better than that of CR2N emphasizing that CG is more conservative and tends to reuse most retrieved keywords. After an initial dip, CR2N’s recall results remain mostly constant. The initial decline in CR2N’s recall is attributed to similar problems in the dataset not sharing the same solution keywords though their solutions might have the similar meaning.

Overall, the retrieval accuracy, precision and recall results obtained are comparatively low in this domain (values are less than 0.5). A closer look suggests that values are misleading as regards the actual effectiveness of the TCBR system. This is because quantitative measures used in our evaluation only count matching keywords using their stems, lemma or synonyms. Therefore, they are unable to capture sentences that have similar meanings when expressed by a slightly different set of keywords. Poor accuracy results are also reported when the retrieved solutions are more detailed than the actual. Table II shows three incident queries as well as the retrieved case, similarity value and retrieval accuracies. With query 1, although the retrieved and actual solutions are similar in meaning, retrieval accuracy is calculated as just 0.333. This is because 1 out of 3 keywords (“nurse/nursing”) is matched in the retrieved solution. Query 3

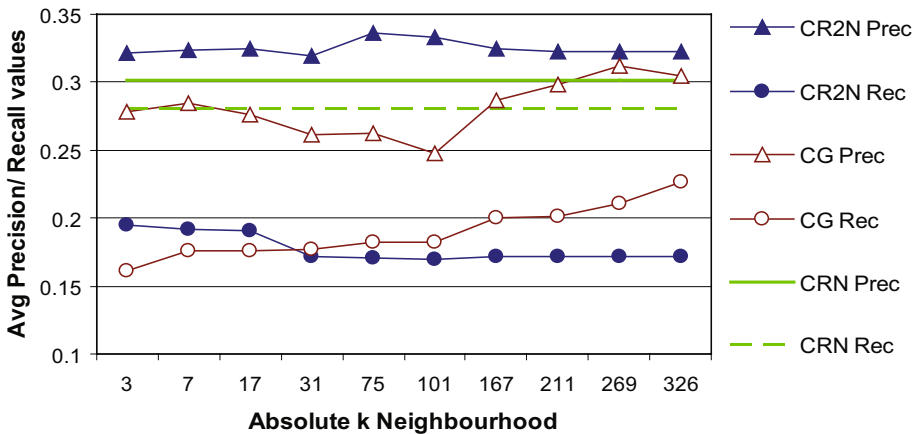


Fig. 8. Precision/Recall results for CR2N, CG and CRN in H&S incident reporting

**Table 1.** Sample retrievals from the Health & Safety incident reports

	Query	Retrieved Problem	Similarity	Retrieved Solution	Actual Solution	Retrieval Accuracy
1	nurse slipped and fell on wet floor	staff member slid on something wet and fell to the floor	0.61237	examined by nursing staff	nurse given first aid	0.333
2	patient fell to the ground as Nurse assisted him to bed.	patient fell out of bed.	0.7071	examined by medical staff.	Patient was advised to get assistance in and out of bed.	0.0
3	Needlestick injury sustained.	needlestick injury sustained by a member of staff.	0.7746	first aid, blood sample taken, visited occupational health.	occupational health contacted.	0.333

poses a similar challenge whilst query 2 highlights a slightly different problem. Here, the omission of information (the fact that the patient would have been examined first) caused the accuracy to be calculated as 0.0. These examples demonstrate the challenges posed by variability in vocabulary and the need for semantics-aware evaluation metrics for TCBR.

## 6 Conclusions and Future Work

The contribution of this work is two fold. Firstly, it proposes the reuse stage in TCBR as identification of reusable textual constructs in a retrieved solution text; the similarity assumption is used to determine reusable constructs. This is then followed by the revision of constructs that have been deemed to be non-reusable. Secondly, it provides an integration of the retrieval and reuse stages in TCBR into a single architecture called CR2N.

Three issues of *when*, *what* and *how* to revise need to be addressed when revising a piece of text. CR2N introduced in this paper addresses the issue of *what* to revise at the reuse stage by automatically annotating components of a solution text as reuse or revise. This is done by extending the CRN architecture and obtaining evidence for reuse/revise from neighbouring cases in the solution space. Experiments with CR2N on two datasets from the domains of weather forecast revision and health & safety incident reporting show better accuracy over a comparable reuse technique (CG) and a retrieve-only system (baseline).

We intend to improve CR2N by capturing context (e.g. influence of left and right adjacent keywords) for each keyword in the CReuseNet and to experiment with other levels of text granularity such as phrases and sentences. A qualitative evaluation (human validation) of our technique is needed to address problems encountered with quantitative evaluation on the health and safety incident report. We also intend to experiment with compositional text reuse where k-nearest cases of a query are combined after identifying reusable keywords by our technique.

**Acknowledgements.** This research work is funded by the Northern Research Partnership (NRP) and the UK-India Education and Research Initiative (UKIERI).

## References

1. Brüninghaus, S., Ashley, K.D.: Reasoning with textual cases. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS, vol. 3620, pp. 137–151. Springer, Heidelberg (2005)
2. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AICom* 7, 39–59 (1994)
3. Plaza, E., Arcos, J.L.: Constructive adaptation. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416, pp. 306–320. Springer, Heidelberg (2002)
4. Gervás, P., Díaz-Agudo, B., Peinado, F., Hervás, R.: Story plot generation based on CBR. In: Twelveth Conference on Applications and Innovations in Intelligent Systems. Springer, Heidelberg (2004)
5. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Textual cbr in jcolibri: From retrieval to reuse. In: Proceedings of the ICCBR 2007 Workshop on Textual CBR: Beyond Retrieval, pp. 217–226 (2007)
6. Lamontagne, L., Lapalme, G.: Textual reuse for email response. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 234–246. Springer, Heidelberg (2004)
7. Lenz, M., Burkhard, H.D.: Case retrieval nets: Basic ideas and extensions. In: Görz, G., Hölldobler, S. (eds.) KI 1996. LNCS, vol. 1137, pp. 227–239. Springer, Heidelberg (1996)
8. Lenz, M., Burkhard, H.D.: Case retrieval nets: Foundations, properties, implementations and results. Technical report, Humboldt University, Berlin (1996)
9. Chakraborti, S., Lothian, R., Wiratunga, N., Orecchioni, A., Watt, S.: Fast case retrieval nets for textual data. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS, vol. 4106, pp. 400–414. Springer, Heidelberg (2006)
10. Adeyanju, I., Wiratunga, N., Lothian, R., Sripada, S., Craw, S.: Solution reuse for textual cases. In: 13th UK Workshop on Case-Based Reasoning, pp. 54–62. CMS Press, University of Greenwich (2008)
11. Brüninghaus, S., Ashley, K.D.: Evaluation of textual cbr approaches. In: Proceedings of the AAAI 1998 Workshop on Textual CBR, pp. 30–34. AAAI Press, Menlo Park (1998)
12. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J.A., Sánchez, A.: Building cbr systems with jcolibri. Special Issue on Experimental Software and Toolkits of the Journal Science of Computer Programming 69, 68–75 (2007)
13. Sripada, S.G., Reiter, E., Hunter, J., Yu, J.: Sumtime-meteo: Parallel corpus of naturally occurring forecast texts and weather data. Technical Report AUCS/TR0201, Department of Computer Science, University of Aberdeen (2002)
14. Massie, S., Wiratunga, N., Craw, S., Donati, A., Vicari, E.: From anomaly reports to cases. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 359–373. Springer, Heidelberg (2007)
15. Mudambi-Ananthasayanam, R., Wiratunga, N., Chakraborti, S., Massie, S., Khemani, D.: Evaluation measures for TCBR systems. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 444–458. Springer, Heidelberg (2008)
16. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998), <http://wordnet.princeton.edu>

# Case-Based Reasoning in Transfer Learning

David W. Aha<sup>1</sup>, Matthew Molineaux<sup>2</sup>, and Gita Sukthankar<sup>3</sup>

<sup>1</sup> Navy Center for Applied Research in Artificial Intelligence,  
Naval Research Laboratory (Code 5514), Washington, DC 20375

<sup>2</sup> Knexus Research Corporation, Springfield, VA 22153

<sup>3</sup> School of Electrical Engineering and Computer Science,  
University of Central Florida, Orlando, FL 32816

david.aha@nrl.navy.mil, matthew.molineaux@knexusresearch.com,  
gitaras@eecs.ucf.edu

**Abstract.** Positive transfer learning (TL) occurs when, after gaining experience from learning how to solve a (source) task, the same learner can exploit this experience to improve performance and/or learning on a different (target) task. TL methods are typically complex, and case-based reasoning can support them in multiple ways. We introduce a method for recognizing intent in a source task, and then applying that knowledge to improve the performance of a case-based reinforcement learner in a target task. We report on its ability to significantly outperform baseline approaches for a control task in a simulated game of American football. We also compare our approach to an alternative approach where source and target task learning occur concurrently, and discuss the tradeoffs between them.

## 1 Introduction

There is a huge body of research on the study of transfer in psychology and education (e.g., Thorndike & Woodworth, 1901; Perkins & Salomon, 1994; Bransford *et al.*, 2000), among other disciplines. Transfer is, more specifically, a focus of some research related to case-based reasoning (CBR), namely psychologically plausible theories of analogical transfer (e.g., Gentner, 1993; Hinrichs & Forbus, 2007). Arguably, case-based reasoning *is* the study of computational models for knowledge transfer, which plays a central role in the crucial topic of lifelong learning. Given this, there are surprisingly few publications on this topic in the CBR literature today.

Recently, this changed due to an increased emphasis on the study of Transfer Learning (TL), motivated in part by a DARPA research program of the same name.<sup>1</sup> TL concerns the study of how learning to solve tasks in a *source* domain can impact an agent's ability to learn to solve tasks from a *target* domain, where the focus is on *positive* transfer (i.e., involving improvements on measures of task performance). For example, we would expect that knowledge obtained from learning how to play one real-time strategy game (e.g., AGE OF EMPIRES) may assist us in learning to play a related, but different, game in the same genre (e.g., EMPIRE EARTH).

---

<sup>1</sup> [http://www.darpa.mil/ipto/programs/tl/docs/TL\\_Brief.ppt](http://www.darpa.mil/ipto/programs/tl/docs/TL_Brief.ppt)

In this paper, we provide an overview of Transfer Learning and CBR techniques for achieving TL. We also introduce a method that uses *intent recognition* to create an abstraction of an actor’s intentions to facilitate the TL process. This could enable the learner to reformulate problem instances involving the same actor. We conceptualize the actor’s behavior as being governed by a set of hidden state variables that can be discovered through intent recognition and utilized by the transfer learner.

In the following sections, we discuss transfer learning, how CBR has served and can serve in some TL methods, introduce and evaluate a novel method that uses intent recognition to facilitate transfer learning for a case-based reinforcement learner performing a target task, and examine the tradeoffs of TL approaches versus a concurrent learning method (i.e., that gains experience from the source *and* target tasks, simultaneously) (Molineaux *et al.*, 2009). Our study involves tasks drawn from an American Football simulator (RUSH 2008). We will conclude with future research suggestions at the intersection of CBR, TL, and plan recognition.

## 2 Transfer Learning and Case-Based Reasoning

Research on machine learning has traditionally focused on tasks in which examples are repeatedly and independently drawn from an identical distribution (i.i.d.) (Simon, 1983). This simplifying assumption is the basis of most ML research to date, as well as most research on case-based learning. In particular, it underlies the mathematical foundations of many existing supervised learning algorithms, and permits the study of their ability to *generalize* from training data.

This contrasts with reality; people frequently leverage their experience gained from one task to improve their performance on different, novel tasks. That is, they perform transfer learning:

**Definition.** *Transfer learning* (TL) is the practice of recognizing and applying knowledge and skills learned from one or more previous tasks to more efficiently and/or effectively learn to solve novel tasks (in new domains).

Thus, the i.i.d. assumption does not (normally) hold in TL tasks. Yet general methods for TL hold the promise for being exceedingly useful; they could dramatically decrease the amount of training/cases required to achieve a given level of problem-solving competence in one domain by successfully employing knowledge obtained from solving problems for different, but related, tasks. This gap has motivated the development of computational models for TL, which has been the focus of recent workshops at NIPS-05, ICML-06, and AAAI-08<sup>2</sup>, in addition to a large number of publications (e.g., Marx *et al.*, 2005; Raina *et al.*, 2006; Shapiro *et al.*, 2008).

To clarify, Figure 1 summarizes a TL method and its evaluation process. The learning agents, tasks, performance metrics, and environments may all differ between the source and target domains. The key object handed off from source to target is the learned knowledge from the source domain. An evaluation of the process typically compares two conditions: the *transfer* condition (depicted by the entirety of Figure 1), in which case a *Mapper* is used to transform knowledge acquired from training on the

---

<sup>2</sup> <http://teamcore.usc.edu/taylor/AAAI08TL/index.htm>

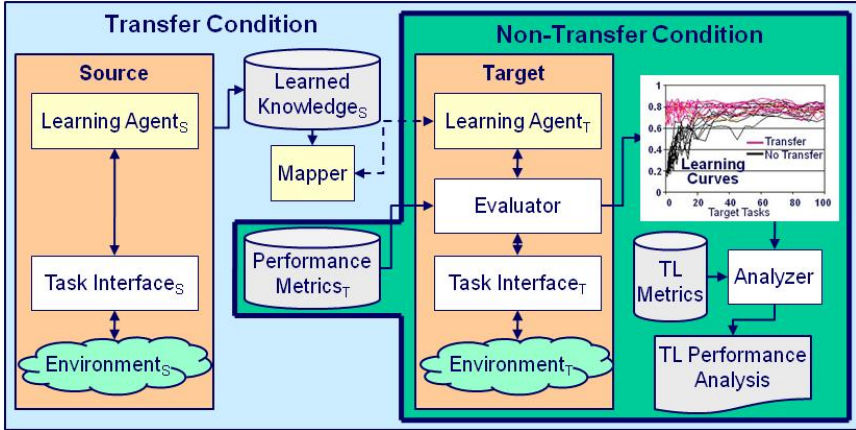


Fig. 1. Transfer learning conditions, method, and evaluation strategy

source task to solve target tasks, and the *non-transfer* condition (depicted as a subsection in Figure 1), in which no such source task knowledge is leveraged. (The dotted line emanating from the Mapper indicates that it is referenced only by the transfer condition.) For these two conditions, two sets of learning curves are generated according to the target task’s performance metric. These, along with a TL metric, are given to an analysis component for significance testing. We discuss examples of these for our study in Section 3.

TL problems vary widely, and a large variety of learning agents have been examined. For example, in some agents, only trivial mapping takes place, while in others mapping is a comprehensive, focal task. In most cases, the mapping is an abstraction designed to assist with relating source and target tasks.

While transfer has been long-studied in CBR (e.g., Goel & Bhatta, 2004), we review only a subset of recent relevant research to provide the context for our contribution. In this context, Table 1 summarizes and distinguishes five branches of prior research. In their TL processes, CBR is used for mapping, for learning in the target domain, or both. For example, Kuhlmann and Stone (2007) enumerate a constrained space of related game descriptions and store graph representations of them as cases. They then use graph isomorphism to map stored cases so as to reuse reinforcement learning (RL) value functions learned from solving source tasks (i.e., related game variants). Liu and Stone (2006) instead use a domain-tailored variant of the structure-mapping engine (SME) (Falkenhainer *et al.*, 1989) to perform source-target task mapping. Their cases encode agent actions in KeepAway soccer games as qualitative dynamic Bayesian networks. The focus is again on value function reuse. Von Hessling and Goel (2005) propose to learn abstractions of Q-learned policies (i.e., as decision trees) indexed by the environment’s features. The policies are to be reused in similar environments. Sharma *et al.* (2007) focus on learning and reusing Q values for state-action pairs, where the feature vector states are case indices representing situations in a real-time strategy game. Their CBR/RL algorithm updates the Q values and eligibility traces of stored cases, where actions denote tactical decisions at the middle level of a three-level action hierarchy.

**Table 1.** Summary of some related work on CBR and transfer learning

Reference	CBR Use	TL Levels	TL Tasks		Environment Type
			Source	Target	
(von Hessling & Goel, ICCBR-05 WS)	Maps <world, decision tree policy> cases/models for Q fn. xfer in RL	2,5	Navigation in Civ. Turn-based Game	Same; Vary map	Turn-Based Strategy: Microworld
(Liu & Stone, AAAI-06)	Map QDBN task models: Modified SME for value fn. xfer in RL	4	KeepAway	Same; Vary # of units	Team Sports: RoboCup Soccer
(Sharma et al., IJCAI-07)	Map case models via Euclidean state sim. for Q fn. xfer for CBR/RL	1,4	Military control	Same; Vary # units or locations	Real-Time Strategy: MadRTS
(Klenk & Forbus, AAAI-07)	Map solutions via knowledge-intensive structure-mapping for CBR	1-6	Mechanics Problem Solving	Several variations!	Question answering: AP Physics
(Kuhlmann & Stone, ECML-07)	Map game description task models via graph isomorphism for RL	3-5	Board games (e.g., mini-chess)	Same; Vary board size, # pieces, etc	General Game Playing: GGP
<b>This Paper</b>	Map feature learned via intent recognition to learn Q fn. for CBR/RL	9	Identify Play of Defense	Control QB in Football Play	Team Sports: Rush 2008

In stark contrast, Klenk and Forbus (2007) do not rely on RL. Instead, they employ SME in a context where cases are solutions to mechanics problems represented in predicate calculus. Given a new problem, they use the candidate inferences generated by SME to construct the necessary model (e.g., applicable equations and assumptions) to arrive at a solution.

Unlike these prior studies, we use an intent recognition approach in the source task to learn a crucial feature (i.e., the defensive team’s play) for use in the target task (i.e., controlling the offensive team’s Quarterback). The source and target tasks in previous work used similar control problems for both tasks, differing, for example, only in the number of agents on the playing field. One way to distinguish transfer learning tasks was suggested in the DARPA TL program, which proposed 11 categories (*levels*<sup>1</sup>) of TL tasks. While intended to be only an *initial* categorization scheme that ranges from simple *Memorizing* (i.e., generalization over i.i.d. examples) to complex *Differing* (i.e., distant-transfer) tasks, it was not subsequently refined by TL researchers. Still, it is useful for our needs: the prior work described above focused on tasks in which, for example, the set of components/units were modified, or the map/formulation differs among the source and target tasks. We instead focused on TL level 9, *Reformulating*, in which a representation transformation is needed to relate the two tasks. This is not to say that our task is more challenging, but rather that it involves different issues.

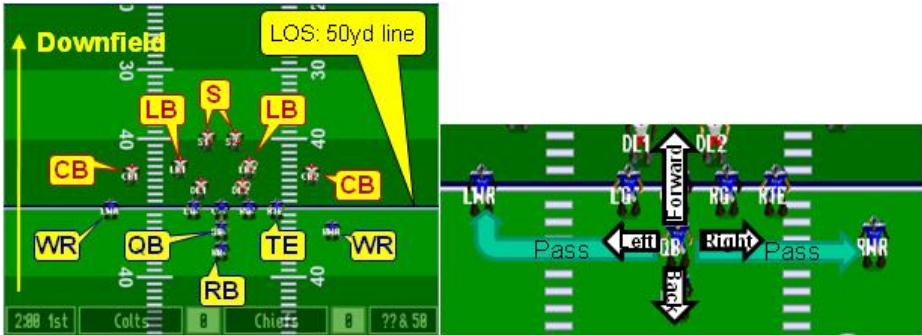
In addition, our task environment differs; it is a multiagent team sports simulator for American football, in which agents differ in their capabilities and pre-determined responsibilities, and the target task concerns controlling a specific agent.

We discuss related work on intent recognition for TL in Section 5. Due to page limitations, please see (Molineaux *et al.*, 2009) for a discussion of related work on case-based reinforcement learning, which is not the focus of our current investigation.

### 3 Case Study: Intent Recognition for Transfer Learning

In this section, we introduce and evaluate a TL agent that performs intent recognition in a source task to improve the performance of a case-based reinforcement learner on a distinct target task. We describe the tasks, the learning algorithms, and their analyses. Our empirical study instantiates the methodology summarized in Figure 1.





**Fig. 2.** Left: The RUSH 2008 starting formation we used for the offense (blue) and defense (red), with some player/position annotations. Right: Six of the QB's eight possible actions. Not shown are **Throw RB** and **Noop**.

### 3.1 Environment, Tasks, and State Representations

Our source and target tasks both use the RUSH 2008<sup>3</sup> environment, which we adapted from the open-source RUSH 2005<sup>4</sup> simulator to support intelligent agent studies. RUSH simulates a simplified American football game on a small field (100x63 yards) where both the offensive and defensive teams have only eight players (see Figure 2). The source task concerns supervised intent recognition: learn to predict the play of the defensive team after observing the first few moments of the play. The defense always starts in the same formation and randomly selects one of 8 possible strategies appropriate for his formation. The defensive players act as follows:

**Defensive Lineman (DL) (2 Players):** These line up across the line of scrimmage (LOS) from the OL (see acronyms below) and try to tackle the ball handler.

**Linebacker (LB) (2):** Starting behind the DL, they will blitz the QB or guard a zone of the field or an *eligible* receiver (i.e., the RB, WR1, or WR2).

**Cornerback (CB) (2):** These line up across the LOS from the WRs and guard a player or a zone on the field.

**Safety (S) (2):** These begin 10 yards behind the LOS and provide pass coverage or chase offense players.

The target task, in which the identity of the defensive play is *not* revealed (i.e., is a hidden state variable), is to learn to control the quarterback's actions on repeated executions of a pass play, where the offensive players perform the following actions:

**Quarterback (QB):** Given the ball at the start of each play while standing 3 yards behind the center of the LOS, our QB agent decides whether and when to run (in one of four possible directions), stand, or throw (and to which receiver).

<sup>3</sup> <http://www.knexusresearch.com/projects/rush/>

<sup>4</sup> <http://rush2005.sourceforge.net/>

**Running Back (RB):** Starts 3 yards behind the QB, runs to an initial position 7 yards left and 4 yards downfield, then charges straight toward the goal line.

**Wide Receiver #1 (WR1):** Starts 16 yards to the left of the QB on the LOS, runs 5 yards downfield and turns right.

**Wide Receiver #2 (WR2):** Starts 16 yards to the right of the QB a few yards behind the LOS, runs 5 yards downfield, and waits.

**Tight End (TE):** Starts 8 yards to the right of the QB on the LOS and pass-blocks.

**Offensive Linemen (OL):** These 3 players begin on the LOS in front of the QB and pass-block (for the QB).

All players are given specific behavioral instructions (i.e., for the offense, a series of actions to execute) except for the QB, whose actions are controlled by our learning agent. The simulator is *stochastic*; each instructed player's actions are random within certain bounds (e.g., the RB will always go to the same initial position, but his path may vary). Each player has varying ability (defined on a 10-point scale) in the categories *power*, *speed*, and *skill*; these affect the ability to handle the ball, block, run, and tackle other players. The probability that a passed ball is caught is a function of the number and skills of defenders near the intended receiver (if any), the skills of the receiver, and the distance the ball is thrown.

RUSH uses a simplified physics; players and the ball each maintain a constant velocity while moving, except that the ball will accelerate downwards due to gravity. Objects are represented as rectangles that interact when they overlap (resulting in a catch, block, or tackle).

For the source task, we found that plays from a given starting formation are usually distinguishable after 3 time steps (Sukthankar *et al.*, 2008). Given this, we use a feature vector representation of length 17 for source task examples. This includes 16 features for the 8 defensive players' displacement in two-dimensional space over the first 3 time steps, plus a label indicating the defensive player's name.

At the start of each play (for both tasks), the ball is placed at the center of the LOS along the 50 yard line. For the target task, the agent's *reward* is 1000 for a touchdown (i.e., a gain of at least 50 yards), -1000 for an interception or fumble, or is otherwise ten times the number of yards gained (e.g., 0 for an incomplete pass) when the play ends. A reward of 0 is received for all actions before the end of the play. Touchdowns (0.01%-0.2%), interceptions, and fumbles (combined: 1%-3%) rarely occur.

For the target task, our learning agent attempts to control the QB's actions so as to maximize total reward. The QB can perform one of eight actions (see Figure 2) at each time step. The first four (**Forward**, **Back**, **Left**, and **Right**) cause the QB to move in a certain direction for one time step. Three more cause the QB to pass to a receiver (who is running a pre-determined pass route): **Throw RB**, **Throw WR1**, and **Throw WR2**. Finally, **Noop** causes the QB to stand still. The QB may decide to run the football himself, and will choose actions until either he throws the ball, crosses into the end zone (i.e., scores a touchdown by gaining 50 yards from the LOS), or is tackled. If the QB passes, no more actions are taken, and the play finishes when an incompletion or interception occurs, or a successful receiver has been tackled or scores a touchdown.

The state representation in the target task contains only two features. The first denotes the defensive strategy (predicted after the third time step), while the second is the time step. For the *transfer* condition, the first feature's value will be predicted using the model transferred from learning on the source task. For the *non-transfer* condition, it will instead be randomly selected, chosen according to a uniform distribution over the defensive plays in use.

### 3.2 Learning Algorithms

Many multi-class supervised learning algorithms can be used for the source task. We considered both support vector machines (SVMs) (Vapnik, 1998), due to their popularity and success, and the k-nearest neighbor classifier (using Euclidean distance), because it is a simple case-based algorithm. A soft-margin SVM for binary classification projects data points into a higher dimensional space, specified by a *kernel* function, and computes a maximum-margin hyperplane decision surface that most nearly separates two classes. Support vectors are those data points that lie closest to this decision surface; if these data points were removed from the training data, the decision surface would change. More formally, given a labeled training set  $T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$  with feature vectors  $\mathbf{x}_i \in \mathfrak{X}^n$  and class labels  $y_i \in \{-1, 1\}$ , a soft margin SVM solves the following to find the maximal hyperplane that (most nearly) separates the two classes:

$$\min_{\mathbf{w}, \mathbf{b}, \xi} = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i$$

constrained by:

$$\begin{aligned} y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + \mathbf{b}) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

where  $\mathbf{w}$  is the normal vector lying perpendicular to the hyperplane,  $\mathbf{b}$  is a bias,  $\xi_i$  is a *slack* variable that measures the degree of misclassification of  $\mathbf{x}_i$ ,  $C$  is a constant, and function  $\phi(\cdot)$  is represented by a kernel function  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i, \mathbf{x}_j) \phi(\mathbf{x}_i, \mathbf{x}_j)$ . We use the radial basis function kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \gamma > 0$$

To work on our 8-class problem, we employ a standard one-vs-one voting scheme where all 28 (i.e.,  $8*(8-1)/2$ ) pair-wise binary classifiers are trained and the most popular class is selected. For our studies, we use the LIBSVM (Chang & Lin, 2001) implementation, and set  $C=1$  and  $\gamma=0.008$ .

In the transfer condition, the model learned from the source task will be provided to the target task learner. The Mapper (Figure 1) will apply this model to the scenario data and output the predicted value of the first (of two) features.

Our recent focus has been on examining the behavior of novel case-based RL algorithms (Molineaux *et al.*, 2008; 2009). Here we use a related algorithm, named *Case-Based Q-Lambda with Intent Recognition* (CBQL-IR), for the target learning task. Based on the  $Q(\lambda)$  algorithm (Sutton & Barto, 1998), it uses a set of case bases to approximate the standard RL  $Q$  function and the trained supervised algorithm from the source task to add opponent intent information to the state.

The Q function approximation maps state-action pairs to an estimate of the long-term reward for taking an action  $a$  in a state  $s$ . There is one  $Q_a$  case base in this set for each action  $a \in \mathbf{A}$ , where  $\mathbf{A}$  is the set of 8 actions defined by the environment. Cases in  $Q_a$  are of the form  $\langle s, v \rangle$ , where  $s$  is a feature vector describing the state and  $v$  is a real-valued estimate of the reward obtained by taking action  $a$  in state  $s$ , then pursuing the current policy until the task terminates. These case bases support a case-based problem solving process (López de Mantaras *et al.*, 2005). At the start of each experiment, each  $Q_a$  case base is initialized to the empty set. Cases are then added and modified as new experiences are gathered, which provide new local estimates of the Q function.

At each time step, a state is observed by the agent, and an action is selected. With probability  $\epsilon$ , a random action will be chosen (*exploration*). With probability  $1-\epsilon$ , CBQL-IR will predict the (estimated) best action to take (*exploitation*). To exploit, it reuses each  $Q_a$  case base by performing a locally-weighted regression using a Gaussian kernel on the retrieved  $k$  nearest neighbors of the current observed state  $s$ . (For faster retrieval, we use  $kd$ -trees to index cases.) Similarity is computed using a normalized Euclidean distance function. This produces an estimate of the value of taking action  $a$  in the current observed state  $s$ . CBQL-IR selects the action with the highest estimate, or a random action if any case base has fewer than  $k$  neighbors.

Once that action is executed, a reward  $r$  and a successor state  $s'$  are obtained from RUSH 2008. This reward is used to improve the estimate of the Q function. If the case is sufficiently novel (greater than a distance  $\tau$  from its nearest neighbor) a new case is retained in  $Q_a$  with state  $s$  and  $v = r + \gamma \max_{a \in \mathbf{A}} Q_a(s')$ , where  $Q_a(\cdot)$  denotes the current estimate for a state in  $Q_a$  and  $0 \leq \gamma < 1$  is the discount factor. If the case is not sufficiently novel, the  $k$  nearest neighbors are revised according to the current learning rate  $\alpha$  and their contribution  $\beta$  to the estimate of the state's value (determined by a normalization over the Gaussian kernel function, summing to 1). The solution value (i.e., reward estimate) of each case is updated using:

$$v = v + \alpha\beta[r + \gamma \max_{a \in \mathbf{A}} Q_a(s') - Q_a(s)].$$

Finally, the solution values of all cases updated earlier in the current trial are updated according to their  $\lambda$ -eligibility:

$$v = v + (\lambda\gamma)^t \alpha\beta[r + \gamma \max_{a \in \mathbf{A}} Q_a(s') - Q_a(s)],$$

where  $t$  is the number of steps between the earlier use and the current update, and  $0 \leq \lambda < 1$  is the trace decay parameter.

### 3.3 Empirical Evaluation

We hypothesized that *transferring an intent recognition model learned from the source task* (i.e., described in Section 3.1) *can significantly increase a learner's ability in the target task*. To investigate this, we applied the algorithms described in Section 3.2 for the transfer and non-transfer conditions. We also tested, as a baseline, a simple non-learning agent on the transfer task to provide additional insight on the difficulty of performing that task.

The methodology we use for testing is shown in Figure 1. We have described the learning agents, environments, the model learned from the source task, our source-target

mapping, and the performance metrics. For our evaluation, we use the Lightweight Integration and Evaluation Toolkit (LIET) as the evaluator. That is, we use LIET, a variant of TIELT<sup>5</sup> (Molineaux & Aha, 2005), to integrate the learning agents with RUSH, and to conduct the evaluation.

For the source task, we varied the amount of training data (between 8 and 40 examples) given to the supervised learners; it wasn't clear, a priori, how much training would suffice to obtain good accuracy for predicting the defensive team's strategy. For the target task, we trained each condition for 100K trials. After every 250 training trials, the performance of the CBQL-IR agent was tested 10 times against each of the defensive strategies. The total reward from all 50 testing trials was averaged to obtain an estimate of average performance at each time step. This evaluation was conducted 20 times to ensure repeatability and obtain statistically significant measures of overall transfer performance.

We use three standard TL metrics to measure transfer performance. The first, *jump start*, measures the difference in performance between the transfer and non-transfer conditions before training begins on the target task. Next, *asymptotic gain* is the same measure, but applied to the end of the curves rather than the beginning. Finally, *k-step regret* measures the increase in performance over the entire learning period, defined as the integral difference between two learning curves, divided by the area of the bounding box that extends from the origin horizontally through the last trial of the curve (or the first trial at which they have both reached asymptotic performance) and vertically to the highest accuracy achieved by either averaged curve. Intuitively, this is a percentage of the maximum performance gain possible between an algorithm that always achieves worst-case performance and an algorithm that always achieves best-case performance.

For the analyzer, we used USC/ISI's implementation of randomized bootstrap sampling<sup>6</sup>, which bins the two sets of curves and repeatedly draws a pseudosample (with replacement) for both sets. Applying a TL metric to each pseudosample yields a distribution, which is used to assess whether the two original sets differ significantly.

### 3.4 Results and Analysis

Both supervised learners on the source task generalized well (i.e., recorded test accuracies above 95%) after training on only 1 example from each of the 8 defensive plays. Therefore, we used the trained SVM or k-nearest neighbor classifier with only 8 training examples to provide the transferred model for the transfer condition.

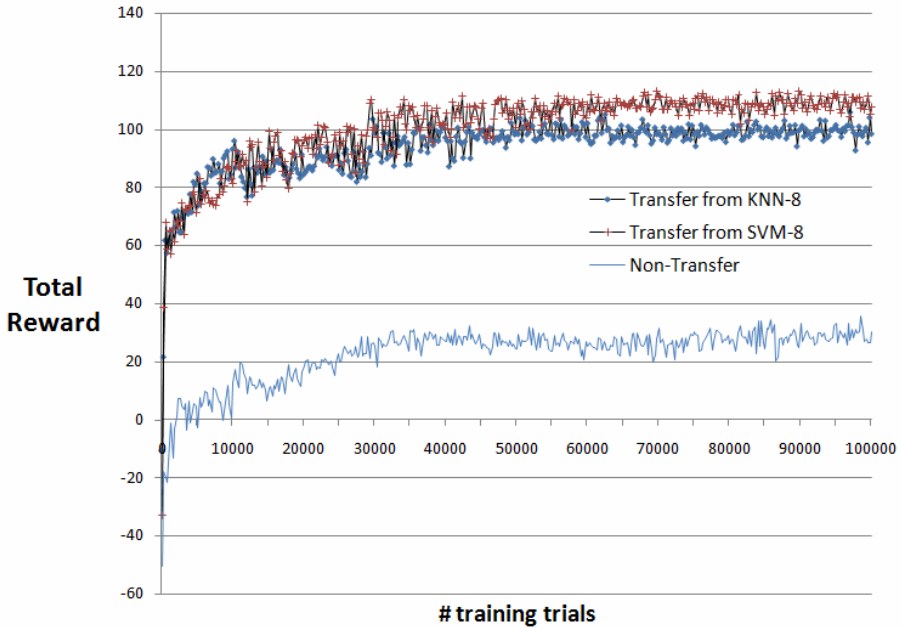
Figure 3 shows the performance on the target task, averaged over the 20 evaluations, for each of three conditions: transfer using a trained support vector machine, transfer using a k-nearest neighbor classifier, and a non-transfer version which predicts any one of the defensive strategies used in the target task with uniform probability. Transfer from 2 and 5 examples of each play yielded performance similar to the kNN and SVM curves depicted, so their performance is not shown. Also not shown is the average performance (over 10K trials) of the non-learning agent on the transfer task because its performance was so low (i.e., -61.1986).

As expected, both transfer variants significantly outperform the non-transfer variant of the agent in terms of both asymptotic advantage and k-step regret. Thus, we

---

<sup>5</sup> <http://www.tielt.org/>

<sup>6</sup> <http://eksl.isi.edu/cgi-bin/page.cgi?page=project-tl-evaluation.html>

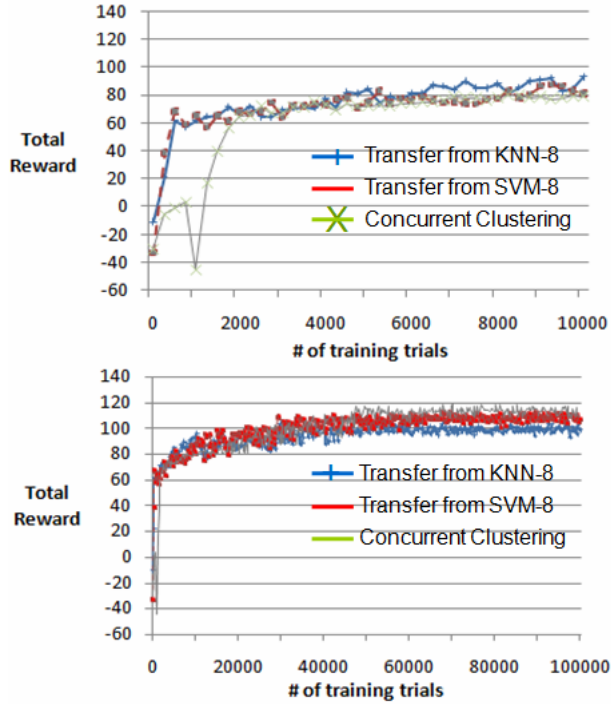


**Fig. 3.** Target task performance on the Quarterback control task for the transfer condition (using either SVM or kNN to learn from the source task), and non-transfer condition’s (Random’s) play predictions

accept our hypothesis as stated in Section 3.3. The regret of the kNN variant over the non-transfer agent is 78.57 (measured over the first 50,000 trials), while the regret of the SVM variant over the non-transfer agent is 78.87 (measured over the first 70,000 trials). The asymptotic advantage of kNN over the non-transfer agent is 59.5; for SVM, it’s 66.66. All of these measures are highly statistically significant with  $p < 0.0001$ . For jump start, there is no statistical advantage between the three curves. This is because CBQL-IR acts randomly when there are no stored cases, as is true before any training occurs; so all three pursue the same policy before training. The differences between SVM and kNN are not statistically significant using either the k-step-regret or asymptotic advantage measures.

## 4 A Concurrent Learning Alternative

Transfer learning is intuitively appealing, cognitively inspired, and has led to a burst of research activity, much of which concerns new techniques involving hierarchical RL. However, transfer is not always cost-effective, and can sometimes result in decreased performance (i.e., *negative* transfer). Also, determining *what* to transfer is a research question itself (e.g., Rosenstein *et al.*, 2005; Stracuzzi, 2006). Yet seemingly overlooked in the literature is a more fundamental question: Should learning be performed separately on the source and target tasks, or can the knowledge learned on the



**Fig. 4.** Target task performance for the SVM and kNN transfer conditions and the concurrent learning (clustering) algorithm (for 10K (top) and 100K (bottom) trials)

source task be instead learned *during* the target task? What are the tradeoffs of doing this versus transfer learning? We found no investigations on this issue.

We briefly address this here, referring to the alternative as *concurrent learning*. In this approach, intent recognition takes place during (rather than prior to) learning to control the QB. As usual during the target task, the label of the defensive play is not given, and must be inferred. We model this as an online unsupervised learning task that clusters the observable movements of the defensive players into groups. The perceived movement  $m \in \mathcal{M}$  for each defensive player is the direction that player is moving during a time step, which has nine possible values:

$$\mathcal{M} = \{None, Forward, Left, Right, Back, Forward-Right, Forward-Left, Back-Right, Back-Left\}$$

These directions are geocentric; *Forward* is always in the direction of play (down-field), and all other directions are equally spaced at  $45^\circ$  angles. Clustering is performed after the third time step of each play, at which time we observe the offsets of the players from their starting formation positions. Thus, 16 features are used to represent defensive plays (i.e., the two-dimensional offsets of each of 8 defensive players). For the first 1000 trials, examples were added to the batch to be clustered, but the predicted cluster (i.e., the recognized play) was *not* used in action selection.

We used the Expectation-Maximization (EM) algorithm from the Weka<sup>7</sup> suite of machine learning software for clustering. EM iteratively chooses cluster centers and builds new clusters until the centers move only marginally between iterations. Also, it increases the number of clusters to discover until successive steps decrease the average log-likelihood of the correct clustering of all points. We selected EM after reviewing several other algorithms; the clusters it found correctly disambiguated the defensive plays over 99% of the time with less than 1000 examples.

Figure 4 compares the performance of the online clustering agent, which learns two tasks concurrently, with the SVM and kNN transfer agents introduced in Section 3. Performance differences are not statistically significant using any of our metrics over the full training period (100,000 trials). However, there is a significant benefit to performing transfer during early learning; transfer achieves good performance far more quickly. As shown, both TL algorithms learn quickly during the first 500 trials, reaching a total reward greater than 60 on average. The concurrent learning algorithm instead takes 2,000 trials to reach the same performance level.

Regret for the kNN TL agent versus the concurrent learning agent for the first 2,000 trials is 29.4, and for the SVM agent is 29.3. Both TL agents statistically outperform the online agent with  $p < 0.0001$  during this early stage of learning.

The concurrent learner's initial learning rate is low due to the time it must spend learning to recognize opponent behaviors, which the TL algorithms learned from the source task. Also, it discovered more clusters than the actual number of defensive plays used. Finally, early examples provided by the CBR/RL agent aren't helpful; they are either too short, or atypical, because that agent does not yet have a successful policy for controlling the QB, whose actions affect the play outcome and the actions of the defensive team. The benefit of TL is that the QB used in the target task is already an expert at "reading" the defense, and so good examples can be obtained starting with the first trial on the target task.

The benefits of TL versus concurrent learning should increase with task complexity because more complex tasks typically require more knowledge to learn, which is often easier to obtain via transfer from simpler tasks. However, there is a cost to TL, both in terms of engineering (in some cases, the source task may be in an entirely different domain) and in higher overall computational complexity. Practitioners interested in multi-task learning may benefit by instead using concurrent learning.

## 5 Discussion: Intent Recognition, TL, CBR, and Future Work

In his seminal work on plan recognition, Kautz (1987) described his event hierarchy circumscription framework as a process for determining "which conclusions are absolutely justified on the basis of the observations, the recognizer's knowledge, and a number of explicit closed-world assumptions." In general, this union of observations, prior knowledge, and closed-world assumptions characterizes research efforts on *plan recognition*. Typically this prior knowledge is encapsulated into a plan library of generative models (either logical (Kautz, 1987) or probabilistic (Bui, 2002)) that the recognizer matches against streams of observations.

---

<sup>7</sup> <http://www.cs.waikato.ac.nz/ml/weka/>



In *intent recognition*, the recognizer attempts to identify useful features relating to an actor's future actions (e.g., the next action to be performed) without an explicit representation of the plan generation model. Even without knowledge of the underlying generative model, being able to discriminate between different types of plans or actions sufficiently in advance can be extremely valuable (e.g., as shown in our study). For instance, in applications involving opponent or user modeling, having limited knowledge of the actor's intentions at an early stage can be more useful than having complete information about the entire plan later in the execution process.

Prior work on TL and plan recognition has focused on the problem of transferring recognition models between different actors. Liao et al. (2005) demonstrated a method for using data from other users to learn priors for a discriminative location-based activity recognition model; an alternate approach is to use prior knowledge to dictate the *structure* of the model rather than the parameters (Natarajan *et al.* 2007). A general issue is that plan generation models based on propositional representations (e.g., the basic HMM) do not generalize well across different users, so there has been work on inference methods for more expressive relational and hierarchical HMM variants that have superior generalization properties (Natarajan *et al.* 2008; Blaylock & Allen, 2006). In our work, the source task (learning a discriminative model of the opponent's play) is quite dissimilar from the target task (learning an optimal single-agent play policy) and we do not address the problem of generalizing across actors.

While several researchers have addressed the topic of CBR and intent or plan recognition, none have proposed its use to facilitate transfer learning. For example, Fagan and Cunningham (2003) analyze a method for predicting a player's actions in a computer game, where the task was supervised learning rather than transfer learning. Kerkez and Cox (2003) represent plans as state-action sequences, index them using an abstract representation (i.e., the number of generalized predicates instantiated in a state), and analyze a case-based algorithm for action prediction for multiple domains. However, no transfer was performed.

Case-based methods that leverage knowledge of intentions and/or plans have significant potential for TL. We demonstrated a simple approach of this type where the beneficiary was a case-based reinforcement learner. Our future work includes investigating our techniques on more comprehensive tasks (e.g., learning to control all offensive players to win an entire football game), including those requiring transfer between different problem domains. More generally, CBR can also be used to map the learned knowledge, and in other roles. An excellent line of future research concerns the study of how case-based algorithms can support continuous learning and planning processes in environments with intentional agents where performance depends on the ability to master multiple tasks, and where reuse can significantly reduce the time required before competent performance emerges.

## 6 Summary

We introduced and evaluated a transfer learning strategy that uses intent recognition to assist a case-based reinforcement learner. It significantly improved task performance for an application involving the control of an agent in a multi-agent team sports environment. Our work is novel in its use of intent recognition for this purpose.

We also briefly examined the typically ignored issue of whether concurrent learning strategies should be considered as an alternative to transfer learning. We discussed some of their tradeoffs, but leave their formal analysis for future research.

Case-based reasoning can play significant roles in transfer learning, yet it has received only a limited amount of attention (e.g., Klenk & Forbus, 2007). This is surprising, given its potential as a focal process for mapping learned knowledge, and reducing overall learning time. In our future work, we will examine how case-based approaches can support lifelong learning in multi-task, multi-agent environments in which knowledge of intentions and (e.g., adversarial) plans (e.g., Sukthankar *et al.*, 2008) can be leveraged to improve performance for decision support applications.

## Acknowledgements

This research was supported by DARPA's Information Processing Techniques Office and the Naval Research Laboratory. Thanks also to Matthew Klenk for his suggestions on this paper.

## References

- Blaylock, N., Allen, J.: Fast hierarchical goal schema recognition. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence, pp. 796–801. AAAI Press, Boston (2006)
- Bui, H.: A general model for online probabilistic plan recognition. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 1309–1318. Morgan Kaufmann, Acapulco (2002)
- Bransford, J.D., Brown, A.L., Cocking, R.R. (eds.): How people learn: Brain, mind, experience, and school. National Academy Press, Washington (2000)
- Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines (2001), <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
- Fagan, M., Cunningham, P.: Case-based plan recognition in computer games. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 161–170. Springer, Heidelberg (2003)
- Falkenhainer, B., Forbus, K.D., Gentner, D.: The structure-mapping engine: Algorithm and examples. *Artificial Intelligence* 41(1), 1–63 (1989)
- Gentner, D.: The mechanisms of analogical learning. In: Buchanan, B.G., Wilkins, D.C. (eds.) Readings in knowledge acquisition and learning: Automating the construction and improvement of expert systems. Morgan Kaufmann, San Francisco (1993)
- Goel, A., Bhatta, S.: Design patterns: A unit of analogical transfer in creative design. *Advanced Engineering Informatics* 18(2), 85–94 (2004)
- von Hessling, A., Goel, A.: Abstracting reusable cases from reinforcement learning. In: Aha, D.W., Wilson, D.C. (eds.) Computer gaming and simulation environments: Proceedings of the ICCBR Workshop (2005); S. Bruninghaus (ed.) Workshop Proceedings of the Sixth ICCBR. DePaul University, Chicago
- Hinrichs, T., Forbus, K.: Analogical learning in a turn-based strategy game. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, pp. 853–858. Professional Book Center, Hyderabad (2007)
- Kautz, H.: A formal theory of plan recognition. Doctoral dissertation, University of Rochester, Rochester, NY (1987)

- Kerkez, B., Cox, M.T.: Incremental case-based plan recognition with local predictions. *International Journal on Artificial Intelligence Tools* 12(4), 413–463 (2003)
- Klenk, M., Forbus, K.D.: Measuring the level of transfer learning by an AP physics problem-solver. In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pp. 446–451. AAAI Press, Vancouver (2007)
- Kuhlmann, G., Stone, P.: Graph-based domain mapping for transfer learning in general games. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenić, D., Skowron, A. (eds.) *ECML 2007. LNCS*, vol. 4701, pp. 188–200. Springer, Heidelberg (2007)
- Liao, L., Fox, D., Kautz, H.A.: Location-based activity recognition using relational Markov networks. In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 773–778. Professional Book Center, Edinburgh (2005)
- Liu, Y., Stone, P.: Value-function-based transfer for reinforcement learning using structure mapping. In: *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pp. 415–420. AAAI Press, Boston (2006)
- López de Mantaras, R., McSherry, D., Bridge, D.G., Leake, D.B., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K.D., Keane, M., Aamodt, A., Watson, I.D.: Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review* 20(3), 215–240 (2005)
- Marx, Z., Rosenstein, M.T., Kaelbling, L.P., Dietterich, T.G.: Transfer learning with an ensemble of background tasks. In: Silver, D., Bakir, G., Bennett, K., Caruana, R., Pontil, M., Russell, S., Tadepalli, P. (eds.) *Inductive Transfer: 10 Years Later: Papers from the NIPS Workshop*, Whistler, BC, Canada (2005), <http://iitrl.acadiau.ca/itws05/>
- Molineaux, M., Aha, D.W.: TIELT: A testbed for gaming environments. In: *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pp. 1690–1691. AAAI Press, Pittsburgh (2005)
- Molineaux, M., Aha, D.W., Moore, P.: Learning continuous action models in a real-time strategy environment. In: *Proceedings of the Twenty-First International FLAIRS Conference*, pp. 257–262. AAAI Press, Coconut Grove (2008)
- Molineaux, M., Aha, D.W., Sukthankar, G.: Beating the defense: Using plan recognition to inform learning agents. To appear in *Proceedings of the Twenty-Second International FLAIRS Conference*. AAAI Press, Sanibel Island (2009)
- Natarajan, S., Bui, H.H., Tadepalli, P., Kersting, K., Wong, W.-K.: Logical hierarchical hidden Markov models for modeling user activities. In: Železný, F., Lavrač, N. (eds.) *ILP 2008. LNCS*, vol. 5194, pp. 192–209. Springer, Heidelberg (2008)
- Natarajan, S., Tadepalli, P., Fern, A.: A relational hierarchical model for decision-theoretic assistance. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) *ILP 2007. LNCS*, vol. 4894, pp. 175–190. Springer, Heidelberg (2008)
- Perkins, D.N., Salomon, G.: Transfer of learning. In: Husen, T., Postelwhite, T.N. (eds.) *International Handbook of Educational Research*, pp. 6452–6457. Pergamon Press, Oxford (1994)
- Raina, R., Ng, A.Y., Koller, D.: Constructing informative priors using transfer learning. In: *Proceedings of the Twenty-Third International Conference on Machine Learning*, pp. 713–720. ACM, Pittsburgh (2006)
- Rosenstein, M.T., Marx, Z., Kaelbling, L.P., Dietterich, T.G.: To transfer or not to transfer. In: Silver, D., Bakir, G., Bennett, K., Caruana, R., Pontil, M., Russell, S., Tadepalli, P. (eds.) *Inductive Transfer: 10 Years Later: Papers from the NIPS Workshop*, Whistler, BC, Canada (2005), [http://iitrl.acadiau.ca/itws05/Papers/ITWS10-RosensteinM05\\_ITWS.pdf](http://iitrl.acadiau.ca/itws05/Papers/ITWS10-RosensteinM05_ITWS.pdf)

- Shapiro, D., Könik, T., O'Rorke, P.: Achieving far transfer in an integrated cognitive architecture. In: Proceedings of the Twenty-Third Conference on Artificial Intelligence, pp. 1325–1330. AAAI Press, Chicago (2008)
- Sharma, M., Holmes, M., Santamaria, J.C., Irani, A., Isbell Jr., C.L., Ram, A.: Transfer learning in real-time strategy games using hybrid CBR/RL. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, Hyderabad, India, pp. 1041–1046 (2007), <http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-168.pdf>
- Simon, H.A.: Search and reasoning in problem solving. *Artificial Intelligence* 21, 7–29 (1983)
- Stracuzzi, D.: Memory organization and knowledge transfer. In: Banerjee, B., Liu, Y., Youngblood, G.M. (eds.) *Structural Knowledge Transfer for Machine Learning: Papers from the ICML Workshop*, Pittsburgh, PA (2006), <http://orca.st.usm.edu/~banerjee/icmlws06/>
- Sukthankar, G., Molineaux, M., Aha, D.W.: Recognizing and exploiting opponent intent in Rush Football (Technical Note AIC-09-062). Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington (2008)
- Sutton, R., Barto, A.: *Reinforcement learning: An introduction*. MIT Press, Cambridge (1998)
- Thorndike, E.L., Woodworth, R.S.: The influence of improvement in one mental function upon the efficiency of other functions (I). *Psychological Review* 8, 247–261 (1901)
- Vapnik, V.: *Statistical learning theory*. Wiley & Sons, New York (1998)

# Toward Modeling and Teaching Legal Case-Based Adaptation with Expert Examples

Kevin Ashley<sup>1</sup>, Collin Lynch<sup>2</sup>, Niels Pinkwart<sup>3</sup>, and Vincent Alevén<sup>4</sup>

<sup>1</sup> Intelligent Systems Program (ISP), Learning Research and Development Center & School of Law, University of Pittsburgh, Pittsburgh, Pennsylvania, USA  
ashley@pitt.edu

<sup>2</sup> ISP, University of Pittsburgh, Pittsburgh, Pennsylvania, USA  
collinl@cs.pitt.edu

<sup>3</sup> CSI, Clausthal U. of Technology, Clausthal, Germany  
niels.pinkwart@tu-clausthal.de

<sup>4</sup> Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA  
aleven@cs.cmu.edu

**Abstract.** Studying examples of expert case-based adaptation could advance computational modeling but only if the examples can be succinctly represented and reliably interpreted. Supreme Court justices pose hypothetical cases, often adapting precedents, to evaluate if a proposed rule for deciding a problem needs to be adapted. This paper describes a diagrammatic representation of adaptive reasoning with hypothetical cases based on a process model. Since the diagrams are interpretations of argument texts, there is no one “correct” diagram, and reliability could be a challenge. An experiment assessed the reliability of expert grading of diagrams prepared by students reconstructing examples of hypothetical reasoning. Preliminary results indicate significant areas of agreement, including with respect to the ways tests are modified in response to hypotheticals, but slight agreement as to the role and import of hypotheticals. These results suggest that the diagrammatic representation will support studying and modeling the examples of case-based adaptation, but that the diagramming support needs to make certain features more explicit.

**Keywords:** Case-based adaptation, Hypothetical reasoning, Legal reasoning.

## 1 Introduction

A repository of transcripts of human experts solving problems through case-based adaptation can be a valuable resource for CBR research as a modeling and teaching tool. Given the continuing dearth of empirical data about how humans modify cases to solve problems [24], this resource could support developing computational models of and teaching case-based adaptation [9, p. 7].

The oral arguments of the U.S. Supreme Court (SCOTUS) are one such repository. The transcripts record the use by human expert decision-makers (i.e., the Justices) of case-based reasoning to explore a space of possible solutions as they respond to the

recommendations urged by advocates. Each transcript is an extended argument about how to decide the case in the form of a “multilogue” between one advocate at a time and the nine Justices. The arguments are inherently case-based and include proposing tests (i.e., rules) for deciding the case, drawing analogies to past cases (i.e., precedents), justifying the analogies in terms of principles and policies underlying the legal domain, challenging the proposed tests by posing hypothetical cases and responding to the hypotheticals, for instance, by modifying the proposed test [3].

Designing the hypotheticals and modifying the tests are a kind of case-based adaptation that we hope to study empirically and to model computationally. In order to flesh out a model for generating hypotheticals and adapting tests, we need more, and more detailed, SCOTUS examples and a way to represent them. For our LARGO program, we developed a diagrammatic representation capturing arguments involving hypothetical reasoning in a succinct way that is partially interpretable by the program.

Given our modeling and pedagogical goals, it is important that the argument diagrams are interpretable in a reliable way. Humans must be able to understand and evaluate argument diagrams reliably in order to model the examples, especially if the diagrams will someday be an input/output medium for a program that instantiates the computational model. Inter-rater reliability in interpreting the argument diagrams is, therefore, a precondition for making further progress in modeling and teaching.

Since the diagrams are interpretations of argument texts, however, there will not be a single “correct” diagram. These are complex real examples of case-based adaptation, expressed in text of which the diagrams are interpretations. The texts may be incomplete, and even if not, multiple reasonable interpretations of the texts are normal. Legal problems are ill-defined; there is no one right answer but often competing reasonable arguments employing different interpretations of open-textured terms. That is the reason hypothetical reasoning is important as a technique for dealing with open textured legal terms. This implies that reasonable people may differ as to the description of the role and import of a hypothetical or the level of abstraction with which to formulate the proposed tests.

Thus, it is an empirical question whether the diagrams can be interpreted reliably. An experiment assessed the reliability of expert grading of diagrams prepared by students as they reconstructed examples of hypothetical reasoning in SCOTUS oral arguments. This paper presents preliminary results with respect to inter-rater reliability. In Section 2, we present an example of hypothetical reasoning that highlights the case-based adaptation and a process model of hypothetical argument that provides a high-level account of it. Section 3 relates the current work to previous work on case-based adaptation and reasoning with examples and hypotheticals. Section 4 illustrates the diagrammatic representation of the same example using our LARGO program, an intelligent tutoring system (ITS) designed to teach law students the process of hypothetical argument. The experiment to assess the reliability of interpreting LARGO diagrams of hypothetical reasoning is described in Section 5, where the results are presented and discussed. Conclusions follow in Section 6.

## 2 Reasoning with Hypothetical Cases and Adaptation

The resolution of a case before a court may be subject to conflicting legal principles. The resolution comprises: (1) a result (e.g., the winner is the party that brings suit, the

plaintiff, or the opponent against whom suit is brought, the defendant); (2) a rule that generates that result when applied to the case facts; and (3) a justification of the result and the rule as consistent with precedents and principles/policies.

Hypothetical reasoning involves generating and testing a rule for deciding the dispute. The proposed test is a hypothesis about how to decide the case in the form of rule the advocate proposes and defends as consistent with past cases and underlying principles/policies. A hypothetical is an imagined case that involves such a hypothesis (i.e., a proposed test) and is designed to explore its meaning or challenge it.

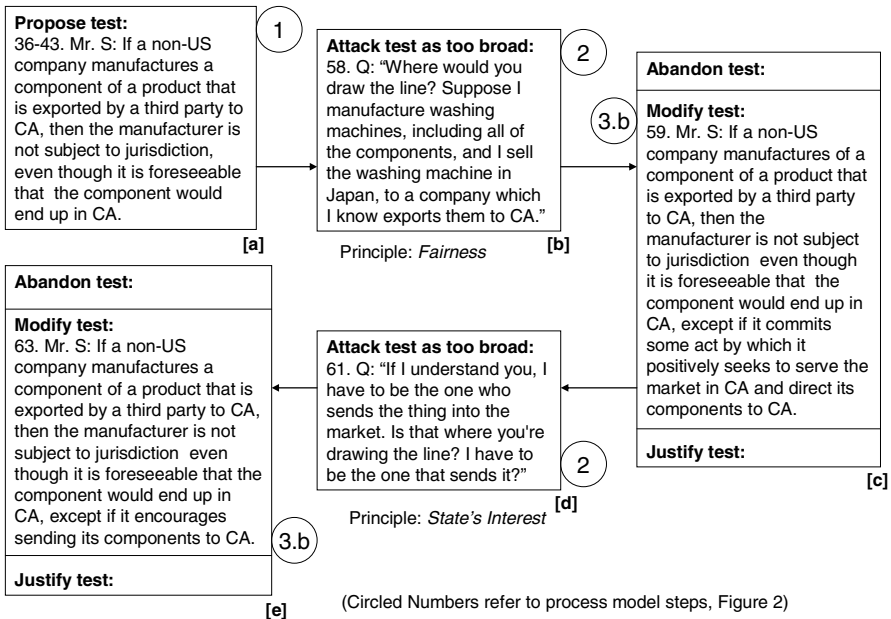
The process of hypothetical reasoning incorporates case-based adaptation, both in the design of an appropriate hypothetical and in the modification of the test. While the hypotheticals are figments of the Justices' imagination, often, they are adaptations of the facts of the current case or past cases. The hypothetical often is designed so that the proposed test applies but reaches a result that contradicts one or more of the underlying principles/policies. That is, the test is too broad. In other situations it is constructed so that the test does not apply but should do so according to one or more of the principles/policies (i.e., the test is too narrow.) In response to the hypothetical, an advocate may adapt the test, narrowing or broadening it as appropriate.

## 2.1 Example of Reasoning with Hypotheticals

We have assembled examples of hypothetical reasoning from a variety of SCOTUS oral arguments, including cases involving freedom of religion, the search warrant requirement, copyright infringement, and, as illustrated here, personal jurisdiction. A standard topic addressed in first year legal process courses, personal jurisdiction refers to the power of a court under the U.S. Constitution to compel a party from outside the state in which the court is located to appear and defend a lawsuit. The underlying legal principles/policies include the due process concern of ensuring fairness to the defendant in requiring him to appear in court within the state versus the state's interest in adjudicating issues and disputes affecting its residents.

The example is based on the petitioner's argument in *Asahi Metal Industry Co. v. Superior Court of California*, 480 U.S. 102 (1987), a case that involved an issue of personal jurisdiction. Specifically, the question is whether Asahi, a Japanese company, may be called into a California court to answer in a civil suit for injuries caused by a blowout of an allegedly faulty motorcycle tire of which Asahi manufactured one component, the tube's valve assembly. Over a fourteen year period, Asahi sold at least 100,000 tire valve assemblies to Cheng Shin, the Taiwanese tube manufacturer. Evidence suggested that Asahi was aware that at least some of its tire valve assemblies would end up in the United States. Furthermore, about twenty percent of the Cheng Shin tires exported to the United States were sold in California.

SCOTUS oral arguments occur after the parties have submitted briefs but before the Justices decide a case or draft an opinion; each side's advocate has one half hour to press his case before the Justices. Since the Court may not be bound to follow the rule in a precedent, or may reinterpret a rule, much of the "action" involves debating about how best to formulate/interpret a rule for deciding the case. For instance, Asahi's advocate, Mr. Staring (Mr. S) argued (in ll. 36-43) that even if it were foreseeable that Asahi's products would end up in California (CA), that was not enough to subject Asahi to the jurisdiction of a CA court. In Fig. 1, box [a] shows an interpretation of Mr. S's point as a proposed test for deciding the case in favor of his client. It is



**Fig. 1.** Posing Hypotheticals to Attack Proposed Test as Too Broad and Modifying Tests

an “interpretation”, because, as frequently occurs in oral argument, the advocate’s and Justices’ positions need to be inferred from what they say; in the oral medium under extreme time pressure, their comments are often very brief.

This proposed test leads the Justice (in box [b], 1. 58 – the line numbers are included to show the proximity of the moves in the transcript) to respond with a hypothetical that suggests the advocate’s test is too broad in defining what would *not* be sufficient for jurisdiction and that a line needs to be drawn somewhere. (The circled numbers in Fig. 1 refer to the steps in the process model of Fig. 2. Posing a hypothetical to challenge the test as too broad is Step 2.) Surely, it would be sufficient if the manufacturer knew that his purchasing exporter was going to send them to California. The change in facts may seem innocuous, but it has a significant effect. In posing the hypothetical, the Justice implies that it clearly would satisfy due process fairness to subject a manufacturer to jurisdiction in a state where he *knew* his product would be shipped. In response, Mr. S. makes his proposed test more specific (in box [c]) by introducing an exception requiring some kind of positive act by the manufacturer to serve the California market. (The exception to the test’s definition of what is *not* subject to jurisdiction acts as a limitation or narrowing of this test.)

The advocate’s narrowing of the test does not sufficiently address one Justice’s concerns about where to draw a line; he worries that the modified test, with its exception requiring some positive act, is still too broad in defining what is not subject to California’s jurisdiction. In box [d], the Justice poses a hypothetical implying that subjecting a manufacturer to a state’s jurisdiction only if he sends the product to the state, limits too severely the state’s interest in enabling its citizens to redress injuries through its courts. In response, box [e], Mr. S. again narrows his test by expanding the



exception somewhat. He concedes that if the manufacturer encourages the sending of the product to California, then it should be subject to California's jurisdiction. At the same time, Mr. S. would maintain, Asahi did nothing to encourage the sending of the tube assemblies to California, distinguishing the case at hand from the hypothetical.

## 2.2 Process Model of Hypothetical Argument

The Process Model of Hypothetical Argument presented in Fig. 2, provides a partial account of the hypothetical reasoning examples we have encountered in SCOTUS oral arguments [3], including the *Asahi* example in Fig. 1. An advocate proposes a test (i.e., a general rule) for deciding the case at hand (step 1 in Fig. 2 illustrated in box [a], Fig. 1). The Justices challenge the proposed test, posing a hypothetical case in order to determine how the proposed test would handle it. Their goal may be to critique the proposed test as too broad (step 2 in Fig. 2, illustrated in boxes [b] and [d], Fig. 1) or too narrow (steps 2' and 3' below the ellipsis, Fig. 2). Alternatively, the Justices may pose a hypothetical case, not in order to critique the test, but simply to explore if the test applies to the hypothetical case and with what result. In modeling this more exploratory use of hypothetical cases, one can relax certain criteria in step 2 or 2', but we do not pursue that here.

In responding at step 3 (or 3'), Fig. 2, the advocate may: (a) stick with his test, justifying it as correctly deciding the case at hand despite the hypothetical, (b) modify the test so that it still assigns the advocate's preferred result in the case but also accommodates the hypothetical, or (c) give up the test and propose another. In the *Asahi* example, Fig. 1, the advocate modifies the test in boxes [c] and [e].

The Process Model incorporates some traditional case-based moves. When responding that the test is not too broad (step 3), justifying the test (3.a) involves analogizing the hypothetical and case; modifying the test (3.b) involves distinguishing the hypothetical from the case. Responding that the test is not too narrow (step 3') involves just the reverse: distinguishing in 3'.a and analogizing in 3'.b. The analogizing involves pointing out relevant shared facts that are reasons for deciding the case and hypothetical the same way. Distinguishing involves pointing out relevant unshared facts that are reasons for deciding the real and hypothetical cases differently.

Facts are relevant, and thus suitable for analogizing and distinguishing, if and to the extent that they matter given the principles/policies of the law. When case facts connect to the law's and regulations' underlying principles/policies, they justify deciding the case consistently with those principles/policies. These principles/policies embody the goals that laws and legal regulations are designed to achieve, for example, to avoid intentionally-inflicted personal injuries, encourage economic competition, discourage frivolous lawsuits, or protect citizens from arbitrary government power. This last is the goal of the law of personal jurisdiction: the due process concern with fairness protects out-of-state citizens from having to defend themselves in court in states to which they have no substantial connections.

## 2.3 Case-Based Adaptation in the Process Model

The Process Model also incorporates more complex moves involving case-based adaptation. In step 2 (or 2') the hypothetical case is designed to demonstrate that the

- **1. Propose Test: For proponent, propose test for deciding the current fact situation (cfs):** Construct a proposed test that leads to a favorable decision in the cfs and is consistent with applicable underlying legal principles/policies and important past cases, and give reasons.
- ← **2. Pose Hypothetical: For interlocutor, pose hypothetical example to probe if proposed test is too *broad*:** Construct a hypothetical example that:
- (a) emphasizes some normatively relevant aspect of the cfs and
  - (b) to which the proposed test applies and assigns the same result as to the cfs, but
  - (c) where, given legal principles/policies, that result is normatively wrong in the hypothetical.
- **3. Respond: For proponent, respond to interlocutor's hypothetical showing test too broad:**
- (3.a) Justify the proposed test: Analogize the hypothetical example and the cfs and argue that they both should have the result assigned by the proposed test. *Or*
  - (3.b) Modify the proposed test: Distinguish the hypothetical example from the cfs, argue that they should have different results and that the proposed test yields the right result in the cfs, and add a condition or limit a concept definition so that the narrowed test still applies to the cfs but does not apply to, or leads to a different result for, the hypothetical example. *Or*
  - (3.c) Abandon the proposed test and return to (1) (i.e., construct a different proposed test that leads to a favorable decision in the cfs and is consistent with applicable underlying legal principles/policies, important past cases, and hypotheticals...)
- ...
- ← **2'. Pose hypothetical: For interlocutor, pose hypothetical example to probe if proposed test is too *narrow*:** Construct a hypothetical example that:
- (a) emphasizes some normatively relevant aspect of the cfs, and
  - (b) that normatively should have the same result as the cfs, but
  - (c) to which the test does not apply or assigns a different result.
- **3'. Respond: For proponent, respond to hypothetical example showing test too narrow:**
- (3'.a) Justify the proposed test: Distinguish the hypothetical and the cfs, arguing that they should not have the same result or that they should have the same result but for different reasons. *Or*
  - (3'.b) Modify the proposed test: Analogize the hypothetical example to the cfs, conceding that the result should be the same in each and arguing that the proposed test yields the right result in the cfs, and eliminate a condition or expand a concept definition so that the test applies to both the cfs and the hypothetical example and leads to the same result in each. *Or*
  - (3'.c) Abandon the proposed test and return to (1) (i.e., construct a different proposed test that leads to a favorable decision in the cfs and is consistent with applicable underlying legal principles/policies, important past cases, and hypotheticals...)

**Fig. 2.** Process Model of Hypothetical Argument

test is too broad (or too narrow). Frequently, the seed for the hypothetical lies in the facts of the case at hand (i.e., the cfs) or of a relevant precedent. The Justices appear to focus on some legally relevant aspect and adapt the seed so that the proposed test

applies to the hypothetical and assigns it the same result as the advocate proposes for the case at hand, but where that result would be wrong in light of the underlying legal principles/policies. Similar adaptations occur in step 2', but the hypothetical is designed so that normatively, it should have the same result as the cfs but does not because the proposed test does not apply or assigns a different result.

Case-based adaptation also occurs in step 3.b (or 3'.b), where the advocate responds to the hypothetical by modifying the proposed test. Having distinguished the hypothetical case from the cfs and argued that they should have different results, the advocate adapts the test by adding a condition or limiting a concept definition so that the narrowed test still applies to the cfs but no longer applies to the hypothetical or leads to a different result. Similar adaptations occur in step 3'.b. Having analogized the hypothetical case and cfs, conceding that the result should be the same in each, the advocate broadens the test, eliminating a condition or expanding a concept definition so that the revised test applies. Although the thing that is modified is the test, the adaptation is still clearly case-based. In each step, the modifications are informed and guided by the distinctions or analogies between the case and hypothetical. Since a Justice designed the hypothetical, these analogies and distinctions indicate his concerns; the modifications to the test are designed to allay those concerns.

One sees both kinds of adaptation in the *Asahi* example of Fig. 1. The Justice's first hypothetical, box [b], changes: (1) the manufacturer of a component part into a manufacturer of the whole product; (2) the assumption that it is foreseeable the product would end up in CA into the company's knowing that it will be exported to CA. The first change simplifies the analysis for purposes of argument; any complexities due to the fact that *Asahi* is the manufacturer of only a component part are temporarily set aside. Arguably, there might be some reason for treating component parts manufacturers more leniently. The second change makes a clearer case for finding that it is fair to subject the manufacturer to personal jurisdiction in CA; it is not just foreseeable that his product will end up there, he *knows* it will.

The advocate's two adaptations are also interesting. Through successive broadening of the exception, each narrows the scope of who is not covered by personal jurisdiction. The first exception, Fig. 1, box [c] covers only those who somehow "commit some act [that] positively seeks to serve the market in CA and direct its components to CA." The second, box [e] is broader; one need only encourage the sending of the component to CA. The impetus for the second adaptation is responding to the Justice's "sending" hypothetical, used both to clarify Mr. S's somewhat obtuse "positive act" requirement and to establish a boundary on extending personal jurisdiction. It is as if the Justice said, "You don't really mean to suggest that personal jurisdiction only applies to one who actually sends the product into CA? CA's interest in protecting its citizens extends farther than that, doesn't it?"

As the example suggests, the details of the adaptations are quite subtle and involve the integration of extensive background knowledge, much of which remains implicit. In fact, this may be the reason Justices employ hypothetical reasoning; it is a remarkably succinct (some might say laconic) way of plumbing the complex implications of a proposed rule. The Process Model skims the surface of these subtleties; extending the model depends on studying more examples in greater detail.

### 3 Related Work

Reasoning with hypothetical cases is a staple of SCOTUS arguments and common law decision making [4; 8; 19], American legal education [5; 22; 23 pp. 66, 68, 75], civil law (i.e., continental European) legal reasoning [14, pp. 528f], ethical reasoning [7] and mathematical discovery [10]. The process model of hypothetical argument of Fig. 2 adapts patterns of hypothetical reasoning observed in legal opinions to a dialogue between an advocate and a judge [4, p. 100]. It adapts three common modes of responding to hypotheticals in order to resolve the dissonance created when a proposed test reaches an arguably undesirable result in a hypothetical [5, pp. 120f]. It focuses on accommodating the conflicting underlying principles at stake [7, pp. 221-8]. The model is similar to Lakatos' mathematical reasoning method of proof and refutations [10, p. 50]. SCOTUS oral arguments are working examples of reasoners' employing hypothetical counterexamples as in the artificial Socratic tutorial dialogue Lakatos reconstructed from centuries-long communications of mathematicians.

As noted, the Process Model provides a high-level account for two types of case adaptation, designing the hypothetical and modifying the proposed test; the above sources, however, do not explain how these subtle adaptations are performed. CBR research on adaptation provides some help. Adaptations like the above can be categorized in terms of the adaptation methods and strategies in [9, p. 395]. Clearly, substitution is involved, but it is a kind of model- and explanation-based substitution based on the knowledge that "knowing" is not only more specific than "foreseeable" but also more strongly supports a legal inference of personal responsibility for the consequences. The other adaptations are based on the knowledge that "sending" is a kind of "positive act" and that "encouraging the sending" is broader than actually sending. Based on other examples, we have suggested the ontological requirements for modeling this kind of model-based substitution with domain facts and factors, legal concepts, principles and policies, and various orderings capturing the kind of legal knowledge illustrated above [2].

A major open question, however, involves the mechanisms to control inferences and moves associated with hypothetical reasoning. The example in Fig. 1 suggests a rhetorical strategy; it shows the beginnings of a slippery slope as the Justice maneuvers Mr. S into needing to explain why supplying component parts to products one knows at least some of which will enter CA is not the very kind of "encouraging the sending" that, according to Mr. S's last test, would subject Asahi to jurisdiction in CA. Some AI research in computationally modeling Lakatos' methods of proof and refutations [6; 15; 12] and reasoning with examples and hypotheticals [1; 20; 21] provides insights into the control problem.

Applying these insights intelligently, however, requires studying many real-world examples. Conducting that kind of empirical study requires a means for adequately representing the examples, namely LARGO diagrams to which we now turn.

### 4 Representing Hypothetical Reasoning Diagrammatically

In order to extend the Process Model to provide a more detailed account of case-based adaptation, to implement the Model computationally, and to teach students this

process of hypothetical reasoning, a succinct representation of the examples is useful. This is especially true since the oral argument examples are described in text, are often distributed across multiple argument “moves” (i.e., turns taken by advocates and Justices), and involve background novel that is only implicit in the transcripts.

We have developed a diagrammatic representation of argument moves involving hypothetical cases, based on our Process Model of Hypothetical Argument [3]. Using the LARGO (Legal ARGument Graph Observer) intelligent tutoring system, students can represent in diagrammatic form portions of SCOTUS oral arguments that relate to hypothetical reasoning [16, 18]. LARGO is intended to help law students learn the process of arguing with hypotheticals by diagrammatically reconstructing examples of SCOTUS oral arguments according to the Process Model. Fig. 3 shows a student’s LARGO diagram representing the same portion of the *Asahi* oral argument discussed in Fig. 1. A scrollable pane (not shown) contains the argument transcript. A student prepared the diagram by selecting and connecting the elements and relations and linking the latter to corresponding passages with a text highlighting feature. There are elements for representing the facts of the case for decision, proposed tests, hypotheticals, and five kinds of relations among them: modifying a test, distinguishing or analogizing a hypothetical, a hypothetical’s leading to a test or modification, and a generic relation. The test element is structured to encourage students to prepare a logical formulation with slots for “if”, “then”, “and”, “unless”, and “even though”.

A somewhat simplified version of the Process Model, together with educationally targeted feedback messages, has been implemented, but a full implementation of the model that would allow the program to make arguments has not been completed. Although LARGO cannot make or respond to hypothetical arguments, it does give advice to students, based on the Process Model, about their argument diagrams. Whenever a student selects the Advice button (not shown), the program provides three new hints on improving the diagram or reflecting on its significance. The advice concerns where to look in the transcript for passages that should be represented in the diagram, how to repair or augment portions of the diagram that appear not to conform to the Process Model, and what patterns of diagram elements appear to be worth reflecting about in terms of the model. In LARGO, a “graph grammar” of rules enforces the expectations embodied in the Process Model. The grammar parses the diagram represented in graph notation [17] in order to flag parts of the diagram where the elements and relations miss relevant parts of the text, do not conform to the Process Model, or are complete enough to warrant reflection.

The graph grammar rules employ classification concepts including a number that focus on CBR functions, for example, distinguishing (or analogizing) without providing reasons, using a general relation between a hypothetical and the cfs rather than analogizing or distinguishing, a hypothetical in isolation (offering an opportunity to enquire if it should connect to a test) and a hypothetical connected to multiple tests (offering an opportunity to discuss if the hypothetical played a role in the modification of one test to another). LARGO’s version of the Process Model does not (yet) explicitly cover the ideas of broadening / narrowing tests and the ways in which hypotheticals are crafted to solicit these test revisions.

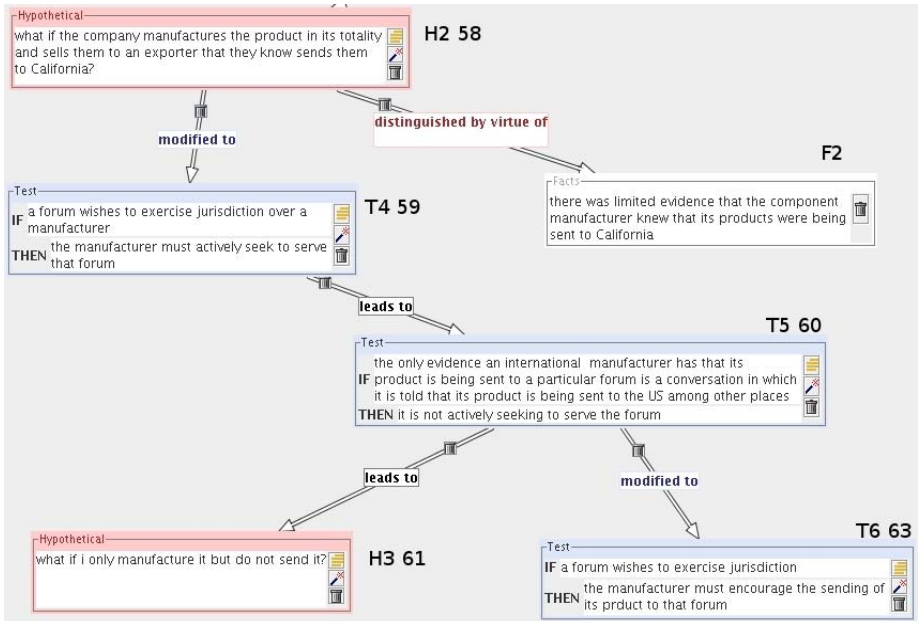


Fig. 3. Sample LARGO Diagram of *Asahi* Oral Argument

LARGO’s advice is couched as a recommendation rather than as a declaration that something is incorrect. The program does not have a “definitive” argument representation; an instructor’s marked-up transcript only indicates where process-model-related components are located in the text. Sometimes, multiple ways of representing argument moves are reasonable, for instance, where different diagrammers interpret the tests at different levels of abstraction. In addition, a Justice may move on to another topic before the advocate can finish; the diagram will be incomplete according to the model but it accurately reconstructs the argument.

The student’s diagram in Fig. 3 satisfies some conventions in the Process Model but violates others. Some relations are mislabeled or omitted. For instance, hypothetical H2 should not be labeled as being modified into test T4. A hypothetical leads to a proposed test’s modification. The student shows that H2 is distinguished from the case facts, but leaves the “distinguished by virtue of” relation unfilled. Where a student has analogized or distinguished a hypothetical as in Fig. 3, LARGO encourages him to explain why this matters (e.g., “Usually, attorneys should give a reason why the distinction matters from a legal viewpoint. For instance, does it matter in terms of the principles and policies underlying the issue? Please enter this in the highlighted distinction relation.”). This student has not done so.

## 5 Experiment to Assess Reliability of Interpreting Diagrams

Given our intention to employ LARGO in computational modeling and teaching, the question is whether humans can interpret the argument diagrams reliably. We have

been comparing argument diagrams created by first and third year law students. The first year students used LARGO as part of a study to determine if the system helped students learn skills of hypothetical reasoning better than a more traditional approach involving reading and note-taking but not diagramming [18]. The third year students used the system in the same ways and context as the first years. In [3] we presented evidence that features of LARGO argument diagrams are correlated with two independent measures related to argumentation ability: standardized test scores that assess ability to evaluate reasoning and arguments and students' number of years in law school. LARGO diagram features, including advice-related classification concepts mentioned above, are also correlated with post-test performance [13].

## 5.1 Experimental Procedure

This experiment involved grading argument diagrams prepared by first and third year law students at the University of Pittsburgh. First year students are typically recent college graduates. Since the third year is the last year of a law school education, it is fair to assume that third year students are more expert in their understanding of legal argument than first years. We used the full set of first diagrams produced by all students who completed the study. This comprised 33 diagrams, prepared in fall 2007 by first year students in their first semester legal process course as part of their regular coursework, and 23 diagrams prepared by volunteer third year students in the middle of their final year. Unlike first-years, the third year students were selected from the top half of their class in terms of law school GPAs and prepared their diagrams for pay outside of class work. The third-years, however, performed the same tasks as the first year students: a pre-test and instruction with LARGO, sessions diagramming three SCOTUS cases, and a post-test, all spread over four two-hour sessions.

Two senior law school professors graded the diagrams following a double-blind procedure. The graders were not aware of whether any diagram was prepared by a first-year or third-year student. The procedure was as follows:

1. Both graders trained on LARGO using the same cases as the students. The graders each produced their own diagrams for the three cases. When grading student diagrams, each grader had his own diagram available.
2. The graders first graded a sample of 6 diagrams drawn from a different study using a draft set of criteria. They graded the diagrams independently and then met to discuss the results and refine the criteria. This ensured that they agreed on and understood the criteria and led to some minor revisions of the criteria.
3. Each grader received the diagrams-to-grade in anonymized form; each diagram had a randomly assigned ID that did not identify the diagrams' author or group. Each grader's diagrams-to-grade were shuffled to ensure that they did not grade them in the same order. Each grader also had the oral argument transcript for which the diagrams were constructed. Annotations on each diagram indicated whether or not an element was linked to the transcript text, and if so to what segment (see Fig. 3, top right lined icon of test and hypothetical elements).
4. Each grader partitioned the diagrams into three bins: poor, medium and good. He then divided each bin into better and worse. This binning resulted in a six-point grading of diagrams based on an initial gestalt inspection. The binning was designed to avoid the reassessment phenomenon in which graders routinely alter their criteria as they grade a set of materials.

**Table 1.** General Grading Criteria and Inter-rater Agreement

Category	Criterion	K
<b>Coverage</b>	How well does the diagram cover ...	
	1. ... all of the essential tests in the argument?	0.05***
	2. ... all of the essential hypotheticals in the argument?	0.75***
	3. ... all of the essential relationships in the argument?	0.62***
	4. How well are the diagram components related to the appropriate facts of the case?	0.56***
	5. ... the argument components as a whole?	0.71***
<b>Correctness</b>	How well does the diagram...	
	1. ... reflect the ways in which the hypotheticals challenge the tests?	0.64***
	2. ... reflect the ways in which tests are modified in response to hypotheticals?	0.69***
	3. ... reflect analogizing and distinguishing of hypotheticals with respect to other hypotheticals and essential case facts?	0.35**
	4. ... capture the role of policies and principles in the argument (e.g., in analogizing and distinguishing)?	0.28**
	5. Overall, how correctly does the diagram represent the argument?	0.7***
<b>Comprehension</b>	How well does the student understand...	
	1. ... this particular argument both in factual and procedural terms?	0.59***
	2. ... the role of proposed tests in legal argument?	0.71***
	3. ... the role of hypothetical cases in argument?	0.09***
	4. ... the process of analogizing and distinguishing hypothetical cases?	0.3*
	5. ... the general process of arguing with tests and hypotheticals?	0.07***
	6. ... the role of policies and principles in arguments of this type?	0.3**

5. Each grader reshuffled the diagrams and (a) assigned detailed grades according to three categories of General Grading Criteria (i.e., coverage, correctness, and comprehension), Table 1; (b) graded each Test and Hypothetical element in the diagram independently according to criteria specific to each type of element, Table 2; and assigned an overall grade to each diagram on a 12 point scale reflecting their by then more complete judgment of the diagram's quality. (One grader assigned overall grades on a 6 point scale. In all of the analyses below, overall grades have been rescaled for comparison.) As Tables 1 and 2 indicate, many of the grading criteria pertain directly to how well student diagrams reflect features of the Process Model of adaptation with hypothetical cases.

## 5.2 Preliminary Results and Discussion

Inter-rater reliability is often measured in terms of the kappa coefficient, which ranges between -1 and 1. How high a kappa value must be to indicate agreement is subject to debate and varies according to the domain, task, and purpose of the grading. Given the lack of domain-specific guidance, we adopted the standards in [11] for strength of agreement for the kappa coefficient:  $\leq 0$ =poor, .01-.20=slight, .21-.40=fair, .41-.60=moderate, .61-.80=substantial, and .81-1=almost perfect.

In analyzing the grades, a comparison of the gestalt rankings using Spearman's Rho, shown in Table 3, line (1), reveals a strong correlation between the graders' scores. As shown in line (2), these rankings were also highly correlated with the graders' final grades, an indication that the more detailed grading process tended to confirm initial assessments rather than alter them. Finally, there was strong inter-grader agreement on the final grades as shown in line (3). For the overall grades we aligned



**Table 2.** Test/Hypothetical Grading Criteria and Inter-rater Agreement

Category	Criterion	$\kappa$
<b>Test Element</b>	1. Is the test summary test like (formulated as a logical rule with applicable conditions and a relevant legal conclusion for deciding an issue or the case)?	0.48***
	2. Is the test linked to an appropriate segment of the argument?	0.02***
	3. Is this test correctly related to the relevant preceding tests?	0.58***
	4. Is this test correctly related to the relevant hypotheticals?	0.62***
	5. How well does the diagram capture the role this test plays in the argument?	0.51***
<b>Hypothetical Element</b>	1. How well does the summary reflect the hypothetical posed in the text?	0.15*
	2. Is this hypothetical correctly related to the relevant test nodes?	0.04***
	3. How well does the diagram capture the role of this hypothetical in the argument with respect to challenging the tests? For instance, does it capture the judge's implication with the hypothetical (i.e., probing the test as too broad, too narrow, or exploring what the test means)?	0.03***
	4. How well does the diagram capture the analogizing and distinguishing of this hypothetical with respect to other hypotheticals and essential case facts?	0.01

**Table 3.** Grader Agreement

Measure	Agreement
(1) Inter-grader ranking agreement	$\rho = 0.71, p < .001$
(2) Intra-grader rank-score agreement	$\kappa = 0.73$ for grader 1, $p < .001$ $\kappa = 0.84$ for grader 2, $p < .001$
(3) Inter-grader score agreement	$\kappa = 0.74, p < .001$

the grades, converting one grader's overall grade to a 12 point scale and correcting the sets to compensate for a difference in mean grades. We then computed agreement using Cohen's weighted kappa with squared weights. Under the standard in [11], the kappa value in Table 3, line (3) indicates "substantial agreement."

The levels of agreement with respect to the General Grading Criteria most relevant to hypothetical reasoning and case-based adaptation vary. There is substantial agreement on coverage of essential hypotheticals (Coverage 2), correctness showing ways hypotheticals challenge tests and ways in which tests are modified in response to hypotheticals (Correctness 1, 3), comprehension of the role of proposed tests (Comprehension 2), and whether the test is correctly related to relevant hypotheticals (Table 2, Test Element 4). There is moderate agreement with respect to whether the diagram captures the role of a test (Table 2, Test Element 5).

There is only fair agreement, however, concerning the correctness and comprehension of how the diagram reflects analogizing and distinguishing hypotheticals or the role of policies and principles (Correctness 3, 4; Comprehension 4, 6). Agreement is slight re: comprehension of the argument role of hypothetical cases and of the general process of arguing with tests and hypotheticals (Comprehension 3, 5), and how well the summary reflects the test and the hypothetical is related to relevant tests, how well the diagram captures the role of the hypothetical, and how well the hypothetical is analogized and distinguished (Table 2, Hypothetical Element 1 – 4).

Generally, these inter-grader agreement results suggest that the diagrams can be interpreted reliably for purposes of instruction and modeling of some aspects of the Process Model, but that the role and import of hypotheticals is problematic. Of course, these are preliminary results, dealing with diagrams of only the first of three cases. The other two sets of diagrams have been graded, but the data are still being

entered and analyzed. Since *Asahi* was the first case graded, the graders may have been uncertain about the criteria associated with the role and import of hypotheticals; the graders may have converged later as they gained practice grading.

In addition, as noted, representing the role and import of hypotheticals is subtle. The diagrams were constructed with our first version of LARGO. We were aware that our tool was unrefined for representing the role of principles/policies in informing analogizing and distinguishing and for representing details about how hypotheticals challenge tests as too broad or narrow. We are exploring the use of pull-down menus with which students annotate the kinds of links between hypotheticals and tests shown in Fig. 3 with information about the role and import of the hypothetical.

## 6 Conclusions

The Supreme Court oral arguments are a repository of examples of hypothetical reasoning and case-based adaptation. A hypothetical case is designed to help evaluate if a test or rule proposed for deciding a problem is consistent with underlying principles/policies and often leads to adaptation of the test to improve consistency. Studying these examples could advance computational modeling of case-based adaptation, especially inference control, strategic reasoning, and creative design in support of case-based adaptation, and aid in teaching the process. A key requirement for progress in modeling and teaching, however, is a means for succinctly representing these examples in a way that humans can interpret reliably.

This paper has described a diagrammatic representation of hypothetical reasoning based on a process model that explains important features of the oral argument examples. An experiment was undertaken to assess the reliability of expert grading of diagrams prepared by students as they reconstructed examples of hypothetical reasoning in the oral arguments. Preliminary results indicate some significant areas of agreement, including with respect to the correctness of ways tests are modified in response to hypotheticals. With respect to other features associated with case-based adaptation such as the role and import of hypotheticals, agreement was slight. These results suggest that the diagrammatic representation will support studying and modeling the examples of case-based adaptation, but that the diagramming support needs to make certain features more explicit.

The researchers plan to reevaluate the results once grading data for two additional cases are analyzed, and to improve the ways in which the diagrams reflect the role and import of the hypotheticals in arguments. They also plan to computationally model realistic legal arguments involving adaptation with hypotheticals.

**Acknowledgments.** NSF Grant IIS-0412830, Hypothesis Formation and Testing in an Interpretive Domain, supported this work.

## References

1. Ashley, K.: Modeling Legal Argument: Reasoning with Cases and Hypotheticals. MIT Press, Cambridge (1990)
2. Ashley, K.: What a Legal CBR Ontology Should Provide. In: Proceedings of the 22nd Int'l FLAIRS Conf. Case-Based Reasoning Track, Sanibel Island, FL (May 2009)

3. Ashley, K., Lynch, C., Pinkwart, N., Alevén, V.: A Process Model of Legal Argument with Hypotheticals. In: *Legal Knowledge and Info. Sys., Proc. Jurix 2008*, pp. 1–10 (2008)
4. Eisenberg, M.: *The Nature of the Common Law*. Harvard U. Press (1988)
5. Gewirtz, P.: The Jurisprudence of Hypotheticals. *J. of Legal Education* 32, 120–124 (1982)
6. Hayes-Roth, R.: Using proofs and refutations to learn from experience. In: Michalski, R., et al. (eds.) *Machine Learning: An A. I. Approach*, Tioga, Palo Alto, pp. 221–240 (1983)
7. Hurley, S.: Coherence, Hypothetical Cases, and Precedent. *Oxford J. Legal Studies* 10, 221–251 (1990)
8. Johnson, T.: *Oral Arguments and Decision Making on the U. S. Supreme Court*, SUNY (2004)
9. Kolodner, J.: *Case-Based Reasoning*. Morgan Kaufmann, San Mateo (1993)
10. Lakatos, I.: *Proofs and Refutations*. Cambridge University Press, London (1976)
11. Landis, J., Koch, G.: The measurement of observer agreement for categorical data. *Biometrics* 33, 159–174 (1977)
12. Lenat, D.B., Brown, J.S.: Why AM and EURISKO appear to work. *Artificial Intelligence* 23(3), 269–294 (1984)
13. Lynch, C., Pinkwart, N., Ashley, K., Alevén, V.: What do argument diagrams tell us about students aptitude or experience? In: *Workshop on ITSs for Ill-structured Domains, ITS 2008*, Montreal (2008)
14. MacCormick, D., Summers, R. (eds.): *Interpreting Precedents Ashgate/Dartmouth* (1997)
15. Pease, A., Colton, S., Smaill, A., Lee, J.: Lakatos and Machine Creativity. In: *Proceedings of the ECAI Creative Systems Workshop* (2002)
16. Pinkwart, N., Alevén, V., Ashley, K., Lynch, C.: Evaluating legal argument instruction with graphical representations using LARGO. In: *Proc. AIED 2007* (July 2007)
17. Pinkwart, N., Ashley, A., Alevén, V., Lynch, C.: Graph Grammars: an ITS Technology for Diagram Representations. In: *Proc. 21st Int'l FLAIRS Conf., ITS Track*, Coral Gables (May 2008)
18. Pinkwart, N., Lynch, C., Ashley, K., Alevén, V.: Re-evaluating LARGO in the Classroom: Are Diagrams Better than Text for Teaching Argumentation Skills? In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) *ITS 2008*. LNCS, vol. 5091, pp. 90–100. Springer, Heidelberg (2008)
19. Prettyman Jr., E.: The Supreme Court's Use of Hypothetical Questions at Oral Argument. *Catholic University Law Review* 33, 555–591 (1984)
20. Rissland, E.: Constrained Example Generation. COINS TR 81-24. U. Mass (1981)
21. Rissland, E.: Dimension-based Analysis of Hypotheticals from Supreme Court Oral Argument. In: *Proc. 2nd Int'l Conf. on Artificial Intelligence and Law*, pp. 111–120. ACM Press, New York (1989)
22. Stuckey, R., et al.: Best Practices for Legal Education, pp. 214–215. *Clin. Leg. Ed. Assc.* (2007)
23. Sullivan, W., Colby, A., Wegner, J., Bond, L., Shulman, L.: *Educating Lawyers*, 62, 66, 68, 75 The Carnegie Foundation for the Advancement of Teaching (2007)
24. Visser, W.: Reuse of Knowledge: Empirical Studies. In: Aamodt, A., Veloso, M.M. (eds.) *ICCBR 1995*. LNCS (LNAI), vol. 1010, pp. 335–346. Springer, Heidelberg (1995)

# Opportunistic Adaptation Knowledge Discovery

Fadi Badra<sup>1</sup>, Amélie Cordier<sup>2</sup>, and Jean Lieber<sup>1</sup>

<sup>1</sup> LORIA (CNRS, INRIA, Nancy Universities)  
BP 239, 54506 Vandœuvre-lès-Nancy, France  
{badra, lieber}@loria.fr

<sup>2</sup> LIRIS CNRS UMR 5202, Université Lyon 1, INSA Lyon, Université Lyon 2, ECL  
43, bd du 11 novembre 1918, Villeurbanne, France  
Amelie.Cordier@liris.cnrs.fr

**Abstract.** Adaptation has long been considered as the Achilles' heel of case-based reasoning since it requires some domain-specific knowledge that is difficult to acquire. In this paper, two strategies are combined in order to reduce the knowledge engineering cost induced by the adaptation knowledge (AK) acquisition task: AK is learned from the case base by the means of knowledge discovery techniques, and the AK acquisition sessions are opportunistically triggered, i.e., at problem-solving time.

## 1 Introduction

Case-based reasoning (CBR [6]) is a reasoning paradigm based on the reuse of previous problem-solving experiences, called cases. A CBR system often has profit of a retrieval procedure, selecting in a case base a source case similar to the target problem, and an adaptation procedure, that adapts the retrieved source case to the specificity of the target problem. The adaptation procedure depends on domain-dependent adaptation knowledge (AK, in the following). Acquiring AK can be done from experts or by using machine learning techniques. An intermediate approach is knowledge discovery (KD) that combines efficient learning algorithms with human-machine interaction.

Most of previous AK acquisition strategies are off-line: they are disconnected from the use of the CBR system. By contrast, recent work aims at integrating AK acquisition from experts to specific reasoning sessions: this *opportunistic* AK acquisition takes advantage of the problem-solving context. This paper presents an approach to AK discovery that is opportunistic: the KD is triggered at problem-solving time.

The paper is organized as follows. Section 2 introduces some basic notions and notations about CBR. Section 3 presents the CBR system TAAABLE, which constitutes the application context of the study, and motivates the need for adaptation knowledge acquisition in this application context. Section 4 presents the proposed opportunistic and interactive AK discovery method. In Sect. 5, this method is applied to acquire adaptation knowledge in the context of the TAAABLE system. Section 6 discusses this approach and situates it among related work. Section 7 concludes and presents some future work.

## 2 Basic Notions About CBR

In the following, problems are assumed to be represented in a language  $\mathcal{L}_{pb}$  and solutions in a language  $\mathcal{L}_{sol}$ . A *source case* represents a problem-solving episode by a pair  $(srce, Sol(srce))$ , in which  $srce \in \mathcal{L}_{pb}$  is the representation of a problem statement and  $Sol(srce) \in \mathcal{L}_{sol}$  is the representation of its associated solution. CBR aims at solving a *target problem*  $tgt$  using a set of source cases  $CB$  called the *case base*. The CBR process is usually decomposed in two main steps: retrieval and adaptation. *Retrieval* selects a source case  $(srce, Sol(srce))$  from the case base such that  $srce$  is judged to be similar to  $tgt$  according to a given similarity criterion. *Adaptation* consists in modifying  $Sol(srce)$  in order to propose a candidate solution  $Sol(tgt)$  for  $tgt$  to the user. If the user validates the candidate solution  $Sol(tgt)$ , then  $Sol(tgt)$  is considered to be a solution  $Sol(tgt)$  for  $tgt$ .

## 3 Application Context: The TAAABLE System

The TAAABLE system [3] is a cooking CBR system. In the cooking domain, CBR aims at answering a query using a set of recipes. In order to answer a query, the system retrieves a recipe in the recipe set and adapts it to produce a recipe satisfying the query. The TAAABLE system was proposed to participate to the *Computer Cooking Contest* (CCC) challenge in 2008 [4]. In the CCC challenge, queries are given in natural language and express a set of constraints that the desired recipe should satisfy. These constraints concern the ingredients to be included or avoided, the type of ingredients (e.g., meat or fruit), the dietary practice (e.g., nut-free diet), the type of meal (e.g., soup) or the type of cuisine (e.g., chinese cuisine). An example of query is: “Cook a chinese soup with leek but no peanut oil.” Recipes are given in textual form, with a shallow XML structure, and include a set of ingredients together with a textual part describing the recipe preparation. The TAAABLE system is accessible online (<http://taaable.fr>).

### 3.1 Representation Issues

*A Cooking Ontology.* The system makes use of a cooking ontology  $\mathcal{O}$  represented in propositional logic. Each concept of  $\mathcal{O}$  corresponds to a propositional variable taken from a finite set  $\mathcal{V}$  of propositional variables.  $\mathcal{O}$  is mainly composed of a set of concepts organized in a hierarchy, which corresponds, in propositional logic, to a set of logical implications  $a \Rightarrow b$ . For example, the axiom  $leek \Rightarrow onions$  of  $\mathcal{O}$  states that leeks are onions.

*Problem and Solution Representation.* In TAAABLE, a problem  $pb \in \mathcal{L}_{pb}$  represents a query and a solution  $Sol(pb)$  of  $pb$  represents a recipe that matches this query.  $\mathcal{L}_{pb}$  and  $\mathcal{L}_{sol}$  are chosen fragments of propositional logic defined using the vocabulary  $\mathcal{V}$  introduced in the cooking ontology  $\mathcal{O}$ . One propositional variable is defined in  $\mathcal{L}_{pb}$  and  $\mathcal{L}_{sol}$  for each concept name of  $\mathcal{O}$  and the only logical connective used in  $\mathcal{L}_{pb}$  and

$\mathcal{L}_{\text{sol}}$  is the conjunction  $\wedge$ . For example, the representation  $\text{tgt} \in \mathcal{L}_{\text{pb}}$  of the query mentioned above is:

$$\text{tgt} = \text{chinese} \wedge \text{soup} \wedge \text{leek} \wedge \neg \text{peanut\_oil}$$

The case base CB contains a set of recipes. Each recipe is indexed in the case base by a propositional formula  $R \in \mathcal{L}_{\text{sol}}$ . For example, the index  $R$  of the recipe *Wonton Soup* is:

$$R = \text{chinese} \wedge \text{soup} \wedge \text{green\_onion} \wedge \dots \wedge \text{peanut\_oil} \wedge \text{Nothing\_else}$$

*Nothing else* denotes a conjunction of negative literals  $\neg a$  for all  $a \in \mathcal{V}$  such that  $\text{chinese} \wedge \text{soup} \wedge \text{green\_onion} \wedge \dots \wedge \text{peanut\_oil} \not\vdash a$ . This kind of “closed world assumption” states explicitly that for all propositional variable  $a \in \mathcal{V}$ , either  $R \vdash a$  (the recipe contains the ingredient represented by  $a$ ) or  $R \vdash \neg a$  (the recipe does not contain the ingredient represented by  $a$ ).

Each recipe index  $R$  represents a set of source cases:  $R$  represents the set of source cases  $(\text{srce}, \text{Sol}(\text{srce}))$  such that  $\text{Sol}(\text{srce}) = R$  and  $\text{srce}$  is solved by  $R$ , i.e.,  $\text{srce}$  is such that  $R \vdash \text{srce}$ .

*Adaptation Knowledge.* In TAAABLE, adaptation knowledge is given by a set of reformulations  $(\mathcal{r}, \mathcal{A}_{\mathcal{r}})$  in which  $\mathcal{r}$  is a binary relation between problems and  $\mathcal{A}_{\mathcal{r}}$  is an adaptation function associated with  $\mathcal{r}$  [13]. A reformulation has the following semantics: if two problems  $\text{pb}_1$  and  $\text{pb}_2$  are related by  $\mathcal{r}$ —denoted by  $\text{pb}_1 \mathcal{r} \text{pb}_2$ —then for every recipe  $\text{Sol}(\text{pb}_1)$  matching the query  $\text{pb}_1$ ,  $\mathcal{A}_{\mathcal{r}}(\text{pb}_1, \text{Sol}(\text{pb}_1), \text{pb}_2) = \widetilde{\text{Sol}}(\text{pb}_2)$  matches the query  $\text{pb}_2$ .

In this paper, binary relations  $\mathcal{r}$  are given by substitutions of the form  $\sigma = \alpha \rightsquigarrow \beta$ , where  $\alpha$  and  $\beta$  are literals (either positive or negative). For example, the substitution  $\sigma = \text{leek} \rightsquigarrow \text{onions}$  generalizes leek into onions.

Adaptation functions  $\mathcal{A}_{\mathcal{r}}$  are given by substitutions of the form  $\Sigma = A \rightsquigarrow B$  in which  $A$  and  $B$  are conjunctions of literals. For example, the substitution  $\Sigma = \text{soup} \wedge \text{pepper} \rightsquigarrow \text{soup} \wedge \text{ginger}$  states that pepper can be replaced by ginger in soup recipes. A substitution  $\Sigma$  can be automatically generated from a substitution  $\sigma$ :  $\Sigma = b \rightsquigarrow a$  if  $\sigma$  is of the form  $a \rightsquigarrow b$  and  $\Sigma = \emptyset \rightsquigarrow \neg a$  if  $\sigma$  is of the form  $\neg a \rightsquigarrow \emptyset$ .

The main source of adaptation knowledge is the ontology  $\mathcal{O}$ . A substitution  $\sigma = a \rightsquigarrow b$  is automatically generated from each axiom  $a \Rightarrow b$  of  $\mathcal{O}$  and correspond to a *substitution by generalization*. A substitution  $\sigma = a \rightsquigarrow b$  can be applied to a query  $\text{pb}$  if  $\text{pb} \vdash a$ .  $\sigma$  generates a new query  $\sigma(\text{pb})$  in which the propositional variable  $a$  has been substituted by the propositional variable  $b$ . For example, the substitution  $\sigma = \text{leek} \rightsquigarrow \text{onions}$  is generated automatically from the axiom  $\text{leek} \Rightarrow \text{onions}$  of  $\mathcal{O}$ .  $\sigma$  can be applied to the query  $\text{tgt}$  to produce the query  $\sigma(\text{tgt}) = \text{chinese} \wedge \text{soup} \wedge \text{onions} \wedge \neg \text{peanut\_oil}$ , in which  $\text{leek}$  has been substituted by  $\text{onions}$ . For each propositional variable  $a$  of  $\mathcal{V}$ , an additional substitution of the form  $\sigma = \neg a \rightsquigarrow \emptyset$  is generated. Such a substitution can be applied to a problem  $\text{pb}$  if  $\text{pb} \vdash \neg a$  and generates a new problem  $\sigma(\text{pb})$  in which the negative literal  $\neg a$  is removed. This has the effect to loosen the constraints imposed on a query e.g., by omitting in the query an unwanted ingredient. For example, the substitution  $\neg \text{peanut\_oil} \rightsquigarrow \emptyset$  applied to

`tgt` generates the query  $\sigma(\text{tgt}) = \text{chinese} \wedge \text{soup} \wedge \text{leek}$ , in which the condition on the ingredient `peanut_oil` is omitted.

However, when  $\mathcal{O}$  is the only source of adaptation knowledge, the system is only able to perform simple adaptations, in which the modifications made to  $\text{Sol}(\text{srce})$  correspond to a sequence of substitutions that can be used to transform `srce` into `tgt`. Therefore, an additional adaptation knowledge base  $\text{AKB}$  is introduced.  $\text{AKB}$  contains a set of reformulations  $(\sigma, \Sigma)$  that capture more complex adaptation strategies.

### 3.2 The CBR Process in TAAABLE

*Retrieval.* The retrieval algorithm is based on a *smooth classification* algorithm on an index hierarchy. Such an algorithm aims at determining a set of modifications to apply to `tgt` in order to obtain a modified query `srce` that matches at least one recipe  $\text{Sol}(\text{srce})$  of the case base. The algorithm computes a *similarity path*, which is a composition of substitutions  $\text{SP} = \sigma_q \circ \sigma_{q-1} \circ \dots \circ \sigma_1$  such that there exists at least one recipe  $\text{Sol}(\text{srce})$  matching the modified query  $\text{srce} = \sigma_q(\sigma_{q-1}(\dots \sigma_1(\text{tgt}) \dots))$ , i.e., such that  $\text{Sol}(\text{srce}) \vDash_{\mathcal{O}} \text{srce}$  holds. Thus, a similarity path  $\text{SP}$  can be written:

$$\text{Sol}(\text{srce}) \vDash_{\mathcal{O}} \text{srce} \xleftarrow{\sigma_q} \xleftarrow{\sigma_{q-1}} \dots \xleftarrow{\sigma_1} \text{tgt}$$

For example, to solve the above query `tgt`, the system generates a similarity path  $\text{SP} = \sigma_2 \circ \sigma_1$ , with:

```
tgt = chinese ∧ soup ∧ leek ∧ ¬peanut_oil
σ1 = ¬peanut_oil ↗ ∅,   σ2 = leek ↗ onions
srce = chinese ∧ soup ∧ onions
Sol(srce) = chinese ∧ soup ∧ green_onion ∧ ... ∧ peanut_oil ∧ Nothing else
```

In this similarity path,  $\text{Sol}(\text{srce})$  is the propositional representation of the recipe *Wonton Soup*. Since the ontology  $\mathcal{O}$  contains the axiom `green_onion`  $\Rightarrow$  `onions`, the modified query  $\text{srce} = \sigma_2 \circ \sigma_1(\text{tgt})$  verifies  $\text{Sol}(\text{srce}) \vDash_{\mathcal{O}} \text{srce}$ .

*Adaptation.* To a similarity path is associated an *adaptation path*  $\text{AP}$ , which is a composition of substitutions  $\text{AP} = \Sigma_1 \circ \Sigma_2 \circ \dots \circ \Sigma_q$  such that the modified recipe  $\widetilde{\text{Sol}}(\text{tgt}) = \Sigma_1(\Sigma_2(\dots \Sigma_q(\text{Sol}(\text{srce})) \dots))$  solves the initial query `tgt`, i.e., verifies  $\widetilde{\text{Sol}}(\text{tgt}) \vDash_{\mathcal{O}} \text{tgt}$ . Thus, an adaptation path  $\text{AP}$  can be written

$$\text{Sol}(\text{srce}) \xrightarrow{\Sigma_q} \xrightarrow{\Sigma_{q-1}} \dots \xrightarrow{\Sigma_1} \widetilde{\text{Sol}}(\text{tgt}) \vDash_{\mathcal{O}} \text{tgt}$$

The adaptation path  $\text{AP}$  is constructed from the similarity path  $\text{SP}$  by associating a substitution  $\Sigma_i$  to each substitution  $\sigma_i$ . To determine which substitution  $\Sigma_i$  to associate to a given substitution  $\sigma_i$ , the external adaptation knowledge base  $\text{AKB}$  is searched first. For a substitution  $\sigma_i = \alpha \rightsquigarrow \beta$ , the system looks for a substitution  $\Sigma = A \rightsquigarrow B$  such that  $A \vDash_{\mathcal{O}} \beta$  and  $B \vDash_{\mathcal{O}} \alpha$ . For example, if  $\sigma_2 = \text{leek} \rightsquigarrow \text{onions}$  is used in  $\text{SP}$  and  $\text{AKB}$  contains the reformulation  $(\sigma, \Sigma)$  with  $\sigma = \sigma_2$  and  $\Sigma = \text{green\_onion} \rightsquigarrow \text{leek} \wedge \text{ginger}$ ,  $\Sigma$  will be selected to constitute the substitution  $\Sigma_2$  in  $\text{AP}$  since

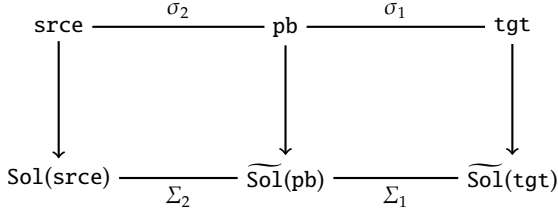


Fig. 1. A similarity path and the associated adaptation path

`green_onion`  $\neq_0$  `onions` and `leek`  $\wedge$  `ginger`  $\neq_0$  `leek`. If no substitution  $\Sigma$  is found in AKB for a given substitution  $\sigma_i$  then  $\Sigma_i$  is generated automatically from  $\sigma_i$ .

In the previous example, AKB is considered to be empty so  $\Sigma_1$  and  $\Sigma_2$  are generated automatically from the substitutions  $\sigma_1$  and  $\sigma_2$ :  $\Sigma_1 = \emptyset \rightsquigarrow \neg\text{peanut\_oil}$  since  $\sigma_1 = \neg\text{peanut\_oil} \rightsquigarrow \emptyset$  and  $\Sigma_2 = \text{onions} \rightsquigarrow \text{leek}$  since  $\sigma_2 = \text{leek} \rightsquigarrow \text{onions}$ . According to the axiom `green_onion`  $\Rightarrow$  `onions` of  $\mathcal{O}$ , the system further specializes the substitution  $\Sigma_2$  into the substitution `green_onion`  $\rightsquigarrow$  `leek` and the user is proposed to replace green onions by leek in the recipe *Wonton Soup* and to suppress peanut oil. The generated adaptation path is  $\text{AP} = \Sigma_1 \circ \Sigma_2$  (Fig. 1), with:

$$\begin{aligned} \text{Sol}(\text{srce}) &= \text{chinese} \wedge \text{soup} \wedge \text{green\_onion} \wedge \dots \wedge \text{peanut\_oil} \wedge \text{Nothing\_else} \\ \Sigma_2 &= \text{green\_onion} \rightsquigarrow \text{leek}, \quad \Sigma_1 = \emptyset \rightsquigarrow \neg\text{peanut\_oil} \\ \widetilde{\text{Sol}}(\text{tgt}) &= \text{chinese} \wedge \text{soup} \wedge \text{leek} \wedge \dots \wedge \neg\text{peanut\_oil} \wedge \text{Nothing\_else} \\ \text{tgt} &= \text{chinese} \wedge \text{soup} \wedge \text{leek} \wedge \neg\text{peanut\_oil} \end{aligned}$$

The inferred solution  $\widetilde{\text{Sol}}(\text{tgt})$  solves the initial query `tgt`:  $\widetilde{\text{Sol}}(\text{tgt}) \neq_0 \text{tgt}$ .

### 3.3 Why Learning Adaptation Knowledge in TAAABLE?

In the version of the TAAABLE system that was proposed to participate in the CCC challenge,  $\text{AKB} = \emptyset$  so adaptation knowledge is inferred from the ontology  $\mathcal{O}$ . The main advantage of this approach lies in its simplicity: no external source of adaptation knowledge is needed and the system is able to propose a solution to any target problem. However, the system's adaptation capabilities (simple substitutions) appear to be very limited and the user has no means to give some feedback on the quality of the proposed adaptation.

For example, the substitution  $\Sigma_1 = \emptyset \rightsquigarrow \neg\text{peanut\_oil}$  suggests to remove the ingredient peanut oil in the retrieved recipe, but as the oil is used in this recipe to saute the bok choy, the adapted recipe turns out to be practically unfeasible. A better adaptation would suggest to replace peanut oil by e.g., sesame oil, which can be modeled by the substitution  $\Sigma_1 = \text{peanut\_oil} \rightsquigarrow \text{sesame\_oil}$ . To generate this substitution automatically, the system could for example exploit the fact that the concepts `peanut_oil` and `sesame_oil` are both sub-concepts of the concept `oil` in  $\mathcal{O}$ . But still, some additional knowledge would be needed to express the fact that peanut oil should be replaced by sesame oil, and not by olive oil or hot chili oil, as `olive_oil` and `hot_chili_oil` are also sub-concepts of `oil` in  $\mathcal{O}$ . Besides, the system should



be aware that this substitution is recommended *only in Asian cuisine*, which can be modeled by the more precise substitution  $\Sigma_1 = \text{asian} \wedge \text{peanut\_oil} \rightsquigarrow \text{asian} \wedge \text{sesame\_oil}$ .

Furthermore, the second substitution  $\Sigma_2 = \text{green\_onions} \rightsquigarrow \text{leek}$  suggests to solely replace sliced green onions by uncooked leek. But the green onion was used in the original *Wonton Soup* for garniture, so the user might consider that raw leek added as garniture alters too much the taste of a soup. A better adaptation would consist in frying leek with e.g., tempeh and red bell pepper to prepare the garniture. Such an adaptation can be modeled by the substitution  $\Sigma_2 = \text{green\_onions} \rightsquigarrow \text{leek} \wedge \text{tempeh} \wedge \text{red\_bell\_pepper}$ . This substitution, which reflects a cooking know-how, can hardly be generated automatically from the ontology.

These examples show that in order to improve its adaptation capabilities, the system would greatly benefit from the availability of a set of adaptation rules that would capture more complex adaptation strategies. These adaptation rules cannot be generated automatically from the ontology and need to be acquired from other knowledge sources. These examples also show that the human expert plays a major role in adaptation knowledge acquisition and that in the cooking domain, adaptation rules are often highly contextual.

## 4 Opportunistic Adaptation Knowledge Discovery

The presented AK acquisition method combines two previous approaches of AK acquisition. The first one was implemented in the CABAMAKA system [5] and learns AK from differences between cases by the means of knowledge discovery techniques (section 4.1). The second one was implemented in the IAKA system [8] and acquires adaptation knowledge at problem-solving time through interactions with the user (section 4.2).

### 4.1 Adaptation Knowledge Discovery from the Case Base

Machine learning algorithms aim at extracting some regularities from a set of observations. Knowledge discovery techniques combine efficient machine learning algorithms with human-machine interaction. In [5], AK is learned from differences between cases by the means of knowledge discovery techniques. A set of pairs of sources cases is taken as input of a frequent itemset extraction algorithm, which outputs a set of itemsets. Each of these itemsets can be interpreted as an adaptation rule. This approach of AK learning was motivated by the original idea proposed by Kathleen Hanney and Mark T. Keane in [11], in which the authors suggest that AK may be learned from differences between cases. The main assumption is that the differences that occur between cases in the case base are often representative of differences that will occur between future problems and the case base.

To learn adaptation rules from differences between cases, representing variations between cases is essential. In [2], expressive representation formalisms are proposed and it is shown that defining a partial order on the variation language can help organizing the learned rules by generality.

## 4.2 Opportunistic and Interactive Knowledge Acquisition

Experiential knowledge, or know-how, can often be acquired on-line, when users are using CBR tools. It is the aim of interactive and opportunistic knowledge acquisition strategies to support such an acquisition. In these strategies, the system exploits its interactions with its user to build new pieces of knowledge, to test them and, in case of success, to retain them. Moreover, the knowledge acquisition process is often opportunistic, i.e., triggered by a previous reasoning failure: reasoning failures highlight missing knowledge and thus constitute a guidance for the acquisition process. A major advantage of interactive knowledge acquisition strategies is that they ensure that the user is in a favorable context when he participates to the acquisition process. In [7], a review of interactive and opportunistic knowledge acquisition approaches is proposed, and two strategies are developed. This work illustrates the efficiency of interactive and opportunistic knowledge acquisition approaches to acquire specific knowledge. On the other hand, it shows that such approaches only allow the systems to acquire small pieces of knowledge at a time.

## 4.3 Combining the Two Approaches

When properly used, knowledge discovery techniques may have the strong advantage of automating a part of the knowledge acquisition process. In these approaches, dedicated human-machine interfaces allow the expert, through predefined interactions, to provide feedback on a set of suggestions generated automatically by the system. The role of the expert is thus reduced to the validation of a pre-selected set of knowledge pieces. The acquired knowledge is directly usable by the system, without the need for an additional formalization step. Automatic approaches also benefit from efficient machine learning algorithms that can be applied, as in [2], to learn adaptation rules at different levels of generality. However, these approaches still produce a large number of candidate knowledge units that have to be validated by a domain expert out of any context, which constitutes an important drawback.

Acquiring adaptation knowledge *offline*, i.e., independently of a particular problem-solving session, appears to be problematic. Offline AK acquisition forces the system's designer to anticipate the need for adaptation knowledge in problem-solving and to acquire it in advance, which can be very tedious, if not impossible. Offline acquisition of adaptation knowledge also makes difficult to come up with fine-grained adaptation rules, since adaptation knowledge is often highly contextual. For example, in the cooking domain, an egg can sometimes be substituted by 100 grams of tofu, but this adaptation rule may be applied only to certain types of dishes, like cakes or mayonnaise, and has proved to be irrelevant in order to adapt a mousse recipe or an omelet recipe. Acquiring such a rule would require to circumscribe its domain of validity in order to avoid over-generalization.

Moreover, initial acquisition of adaptation knowledge prevents the system from learning from experience. A CBR system with fixed adaptation knowledge has no way to improve its problem-solving capabilities, except by retaining in the case base a new experience each time a problem has been solved, as it is usually done in traditional CBR systems [6].

On the other hand, interactive and opportunistic knowledge acquisition approaches heavily rely on the human expert but ensure that the expert is “in context” when validating knowledge units that are to be acquired. Combining knowledge discovery techniques and interactive approaches, as it is proposed here, could overcome one of the limitations of KD by dramatically reducing the number of candidate adaptation rules presented to the expert. By triggering the process in an opportunistic manner, the expert is able to parametrize the KD in order to focus on specific knowledge to acquire in context. The resulting AK discovery process:

- is performed *on-line*, i.e., in the context of a problem-solving session,
- is *interactive* as adaptation knowledge is learned by the system through interactions with its user who acts as an expert,
- is *opportunistic* as it is triggered by reasoning failure, and, consequently, often helps repairing a failed adaptation,
- makes use of knowledge discovery techniques to provide *assistance* to the user in the formulation of new knowledge: the user is presented with a set of suggestions that are generated automatically from the case base.

## 5 Applying Opportunistic AK Discovery to TAAABLE

In this section, an opportunistic AK discovery process is applied to the context of the TAAABLE system.

### 5.1 AK Discovery

In TAAABLE, the AK discovery process consists in learning a set of substitutions from the case base by comparing two sets of recipes.

*The Training Set.* The training set TS is formed by selecting from the case base a set of pairs of recipes  $(R_k, R_\ell) \in CB \times CB$  and by representing for each selected pair of recipes  $(R_k, R_\ell)$  the variation  $\Delta_{k\ell}$  from  $R_k$  to  $R_\ell$ . The choice of the training set TS results from a set of interactions with the user during which he/she is asked to formulate the cause of the adaptation failure and to pick up a repair strategy.

*Representing Variations.* The variation  $\Delta_{k\ell}$  from a recipe  $R_k$  to a recipe  $R_\ell$  is represented in a language  $\mathcal{L}_\Delta$  by a set of properties. Three properties  $a^-$ ,  $a^+$  and  $a^\pm$  are defined in  $\mathcal{L}_\Delta$  for each propositional variable  $a$  of  $\mathcal{V}$ , and  $\Delta_{k\ell} \in \mathcal{L}_\Delta$  contains:

- the property  $a^-$  if  $R_k \models_O a$  and  $R_\ell \not\models_O a$ ,
- the property  $a^+$  if  $R_k \not\models_O a$  and  $R_\ell \models_O a$ ,
- the property  $a^\pm$  if  $R_k \models_O a$  and  $R_\ell \models_O a$ .

For example, if:

$$R_k = \text{chinese} \wedge \text{soup} \wedge \dots \wedge \text{peanut\_oil} \wedge \text{Nothing\_else}$$

$$R_\ell = \text{chinese} \wedge \text{soup} \wedge \dots \wedge \text{olive\_oil} \wedge \text{Nothing\_else}$$

then  $\Delta_{kl} = \{\text{chinese}^-, \text{soup}^-, \text{oil}^-, \text{peanut\_oil}^-, \text{olive\_oil}^+, \dots\}$ , provided that  $\text{peanut\_oil} \not\subseteq \text{oil}$ ,  $\text{olive\_oil} \not\subseteq \text{oil}$ ,  $R_\ell \not\subseteq \text{peanut\_oil}$  and  $R_k \not\subseteq \text{olive\_oil}$ .

The inclusion relation  $\subseteq$  constitutes a partial order on  $\mathcal{L}_\Delta$  that can be used to organize variations by generality: a variation  $\Delta$  is more general than a variation  $\Delta'$  if  $\Delta \subseteq \Delta'$ .

*Mining.* The learning process consists in highlighting some variations  $\Delta \in \mathcal{L}_\Delta$  that are more general than a “large” number of elements  $\Delta_{kl}$  of TS. More formally, let

$$\text{support}(\Delta) = \frac{\text{card} \{\Delta_{kl} \in \text{TS} \mid \Delta \subseteq \Delta_{kl}\}}{\text{card TS}}$$

Learning adaptation rules aims at finding the  $\Delta \in \mathcal{L}_\Delta$  such that  $\text{support}(\Delta) \geq \sigma_s$ , where  $\sigma_s \in [0; 1]$  is a learning parameter called the support threshold. It can be noticed that if  $\Delta_1 \subseteq \Delta_2$  then  $\text{support}(\Delta_1) \geq \text{support}(\Delta_2)$ . The support threshold also has an influence on the number of generated variations. The number of generated variations increases when  $\sigma_s$  decreases. Thus, specifying a high threshold restricts the generation of variations to the most general ones, which can limit the number of generated variations and save computation time but has the effect to discard the most specific ones from the result set.

Each learned variation  $\Delta = \{p_1, p_2, \dots, p_n\} \in \mathcal{L}_\Delta$  is interpreted as a substitution of the form  $A \rightsquigarrow B$  such that:

- $A \not\subseteq a$  and  $B \not\subseteq a$  if  $a^- \in \Delta$ ,
- $A \not\subseteq a$  and  $B \subseteq a$  if  $a^+ \in \Delta$ ,
- $A \subseteq a$  and  $B \subseteq a$  if  $a^- \in \Delta$ .

For example, the variation  $\Delta = \{\text{oil}^-, \text{peanut\_oil}^-, \text{olive\_oil}^+\}$  is interpreted as the substitution  $\Sigma = \text{peanut\_oil} \rightsquigarrow \text{olive\_oil}$ . The conjunct  $\text{oil}$  is not present neither in  $A$  nor in  $B$  since it is useless:  $\text{peanut\_oil} \not\subseteq \text{oil}$  and  $\text{olive\_oil} \not\subseteq \text{oil}$ .

*Filtering.* For a retrieved recipe  $\text{Sol}(\text{srce})$ , the result set can be filtered in order to retain only the substitutions  $\Sigma = A \rightsquigarrow B$  that can be applied to modify  $\text{Sol}(\text{srce})$ , i.e., such that  $\text{Sol}(\text{srce}) \not\subseteq A$ .

*Validation.* Knowledge discovery aims at building a model of reality from a set of observations. But as a model of a part of reality is only valid with respect to a particular observer, any learned substitution has to be validated by a human expert in order to acquire the status of piece of knowledge.

## 5.2 Opportunistic Adaptation Knowledge Discovery

The AK discovery process turns the case base into an additional source of adaptation knowledge. This new source of knowledge is used during a problem-solving session to provide the CBR system with adaptation knowledge “on demand”. A set of variations  $\Delta$  is learned from the case base by comparing two sets of recipes and each learned variation  $\Delta$  is interpreted as a substitution  $\Sigma$  that can be used to repair the adaptation

path  $AP$ . Each learned substitution  $\Sigma$  is presented to the user for validation together with the corrected solution  $\widetilde{\text{Sol}}(\text{tgt})$  resulting from its application. When the user validates the corrected solution, a new reformulation  $(\sigma, \Sigma)$  is added to the adaptation knowledge base  $AKB$  so that the learned substitution  $\Sigma$  can be later reused to adapt new recipes. The  $AK$  discovery process is triggered either during the adaptation phase, to come up with suggestions of gradual solution refinements (see section 5.4 for an example), or during the solution test phase to repair a failed adaptation in response to the user’s feedback (see section 5.5 for an example).

### 5.3 Implementation

To test the proposed adaptation knowledge acquisition method, a prototype was implemented that integrates the  $TAAABLE$  system [3] and the  $CABAMA$  system [5]. The case base contains 862 recipes taken from the CCC 2008 recipe set. The  $TAAABLE$  system is used to perform retrieval and adaptation. The  $CABAMA$  system is used to learn a set of substitutions  $\Sigma$  from the case base from the comparison of two sets of recipes. As in [5], the mining step is performed thanks to a frequent closed itemset extraction algorithm.

### 5.4 A First Example: Cooking a Chocolate Cake

An example is presented to illustrate how the case base is used as an additional source of adaptation knowledge. The  $AK$  discovery process is parametrized automatically and is used to provide assistance to the user by suggesting some gradual refinements for the proposed solution.

1. *Representing the Target Problem.* In this example, the user wants to cook a chocolate cake with baking chocolate and oranges. The target problem is:

$$\text{tgt} = \text{cake} \wedge \text{baking\_chocolate} \wedge \text{orange}$$

In the  $TAAABLE$  interface, the field “Ingredients I Want” is filled in with the tokens `baking_chocolate` and `orange` and the field “Types I Want” is filled in with the token `cake`.

2. *Retrieval.* The retrieval procedure generates the similarity path  $SP = \sigma_1$  in which the substitution  $\sigma_1 = \text{baking\_chocolate} \rightsquigarrow \text{chocolate}$  is generated automatically from the ontology  $\mathcal{O}$  from the axiom  $\text{baking\_chocolate} \Rightarrow \text{chocolate}$ .  $SP$  is applied to  $\text{tgt}$  in order to produce the modified query  $\text{srce} = \text{cake} \wedge \text{chocolate} \wedge \text{orange}$ . The system retrieves the recipe *Ultralight Chocolate Cake*, whose representation  $\text{Sol}(\text{srce})$  is:

$$\text{Sol}(\text{srce}) = \text{cake} \wedge \text{cocoa} \wedge \text{orange} \wedge \dots \wedge \text{Nothing else}$$

Since the ontology  $\mathcal{O}$  contains the axiom  $\text{cocoa} \Rightarrow \text{chocolate}$ ,  $\text{Sol}(\text{srce})$  solves the query  $\text{srce}$ :  $\text{Sol}(\text{srce}) \models_{\mathcal{O}} \text{srce}$ .

3. *Adaptation.*  $AKB$  is assumed to be empty, so to construct the adaptation path  $AP$ , the substitution  $\text{chocolate} \rightsquigarrow \text{baking\_chocolate}$  is generated automatically from  $\sigma_1$ . This substitution is further specialized into the substitution

$\Sigma_1 = \text{cocoa} \rightsquigarrow \text{baking\_chocolate}$ , according to the axiom  $\text{cocoa} \Rightarrow \text{chocolate}$  of  $\mathcal{O}$ . A first solution  $\text{Sol}(\text{tgt})$  is computed by applying to  $\text{Sol}(\text{srce})$  the adaptation path  $\text{AP} = \Sigma_1$ . The user suggests that an ingredient is missing in  $\text{Sol}(\text{tgt})$  but could not identify a repair strategy. An AK discovery is triggered in order to suggest gradual refinements of  $\text{Sol}(\text{tgt})$ .

4. *Choosing the Training Set.* The training set  $\text{TS}$  is chosen from  $\Sigma_1$ : AK is learned by comparing the recipes containing  $\text{cocoa}$  with the recipes containing  $\text{baking chocolate}$ .  $\text{TS}$  is composed of the set of variations  $\Delta_{kl} \in \mathcal{L}_\Delta$  between pairs of recipes  $(R_k, R_l) \in \text{CB} \times \text{CB}$  such that  $\{\text{cocoa}^-, \text{baking\_chocolate}^+\} \subseteq \Delta_{kl}$ .
5. *Mining and Filtering.* A value is given to the support threshold  $\sigma_s$  and the mining step outputs a set of variations. A filter retains only the variations that correspond to substitutions applicable to modify  $\text{Sol}(\text{srce})$ .
6. *Solution Test and Validation.* The user selects the learned variation  $\Delta = \{\text{cocoa}^-, \text{baking\_chocolate}^+, \text{oil}^-\}$  from the result set.  $\Delta$  is interpreted as the substitution  $\Sigma = \text{cocoa} \wedge \text{oil} \rightsquigarrow \text{baking\_chocolate}$ , which suggests to replace  $\text{cocoa}$  by  $\text{baking chocolate}$  in the retrieved recipe and to remove  $\text{oil}$ . The user explains this rule by the fact that  $\text{baking chocolate}$  contains more fat than  $\text{cocoa}$ , and therefore substituting  $\text{cocoa}$  by  $\text{baking chocolate}$  implies to reduce the quantity of fat in the recipe.

Further solution refinements are proposed to the user. The set of learned variations is filtered in order to retain only the substitutions  $\Delta'$  that are more specific than  $\Delta$ , i.e., such that  $\Delta \subseteq \Delta'$ . Among the retained variations is the variation  $\Delta' = \{\text{cocoa}^-, \text{baking\_chocolate}^+, \text{oil}^-, \text{vanilla}^-\}$ , which is interpreted as the substitution  $\Sigma' = \text{cocoa} \wedge \text{oil} \wedge \text{vanilla} \rightsquigarrow \text{baking\_chocolate}$ .  $\Sigma'$  suggests to also remove  $\text{vanilla}$  in the recipe *Ultralight Chocolate Cake*. The user is satisfied with the refined solution  $\text{Sol}(\text{tgt})$  resulting from the application of the adaptation path  $\text{AP} = \Sigma'$  to  $\text{Sol}(\text{srce})$ , so the reformulation ( $\text{baking\_chocolate} \rightsquigarrow \text{chocolate}$ ,  $\text{cocoa} \wedge \text{oil} \wedge \text{vanilla} \rightsquigarrow \text{baking\_chocolate}$ ) is added to the adaptation knowledge base  $\text{AKB}$ .

## 5.5 A Second Example: Cooking a Chinese Soup

A second example is presented in which the AK discovery process is triggered in response to the user feedback in order to repair the adaptation presented in Sect. 3. In this example, the user is encouraged to formulate the cause of the adaptation failure. A repair strategy is chosen that is used to parametrize the AK discovery process.

1. *Representing the Target Problem.* In this example, the target problem  $\text{tgt}$  is:

$$\text{tgt} = \text{chinese} \wedge \text{soup} \wedge \text{leek} \wedge \neg \text{peanut\_oil}$$

In the `TAAABLE` interface, the field “Ingredients I Want” is filled in with the token `leek`, the field “Ingredients I Don’t Want” is filled in with the token `peanut_oil` and the field “Types I Want” is filled in with the tokens `chinese` and `soup`.

2. *Retrieval.* As in Sect. 3, two substitutions  $\sigma_1 = \neg \text{peanut\_oil} \rightsquigarrow \emptyset$  and  $\sigma_2 = \text{leek} \rightsquigarrow \text{onions}$  are generated automatically from the ontology  $\mathcal{O}$ . The

similarity path  $SP = \sigma_2 \circ \sigma_1$  is applied to  $\text{tgt}$  in order to produce the modified query  $\text{srce} = \text{chinese} \wedge \text{soup} \wedge \text{onions}$ . The system retrieves the recipe *Wonton Soup*, whose representation  $\text{Sol}(\text{srce})$  solves the query  $\text{srce}$ :  $\text{Sol}(\text{srce}) \models_{\mathcal{O}} \text{srce}$ .

3. *Adaptation.* Initially,  $\text{AKB} = \emptyset$ , so to construct the adaptation path  $\text{AP}$ , two substitutions  $\Sigma_1 = \emptyset \rightsquigarrow \neg \text{peanut\_oil}$  and  $\Sigma_2 = \text{green\_onion} \rightsquigarrow \text{leek}$  are automatically generated from  $\sigma_1$  and  $\sigma_2$ .
4. *Solution Test and Validation.* The solution  $\widetilde{\text{Sol}}(\text{tgt})$  is presented to the user for validation, together with the adaptation path  $\text{AP} = \Sigma_1 \circ \Sigma_2$  that was used to generate it.
5. *The User is Unsatisfied!* The user complains that the adapted recipe is practically unfeasible because the proposed solution  $\widetilde{\text{Sol}}(\text{tgt})$  does not contain oil anymore, and oil is needed to saute the bok choy.
6. *What has Caused the Adaptation Failure?* The cause of the adaptation failure is identified through interactions with the user. The user validates the intermediate solution  $\widetilde{\text{Sol}}(\text{pb})$  that results from the application of the substitution  $\Sigma_2 = \text{green\_onion} \rightsquigarrow \text{leek}$  to  $\text{Sol}(\text{srce})$ . But the user invalidates the solution  $\widetilde{\text{Sol}}(\text{tgt})$  that results from the application of  $\Sigma_1 = \emptyset \rightsquigarrow \neg \text{peanut\_oil}$  to  $\widetilde{\text{Sol}}(\text{pb})$ . The substitution  $\Sigma_1$  is identified as responsible for the adaptation failure since its application results in the removal of oil in the recipe.
7. *Choosing a Repair Strategy.* A repair strategy is chosen according to the user's feedback. The user expresses the need for oil in the adapted recipe, so the repair strategy consists in replacing peanut oil by another oil. An AK discovery process is triggered to decide which oil to replace peanut oil with.
8. *Choosing the Training Set.* A set of recipes that contain peanut oil is compared with a set of recipes containing other types of oil. The training set  $\text{TS}$  is composed of the set of variations  $\Delta_{kl} \in \mathcal{L}_{\Delta}$  between pairs of recipes  $(R_k, R_l) \in \text{CB} \times \text{CB}$  such that  $\{\text{oil}^-, \text{peanut\_oil}^-\} \subseteq \Delta_{kl}$ .
9. *Mining and Filtering.* A value is given to the support threshold  $\sigma_s$  and the mining step outputs a set of variations. A filter retains only the variations that correspond to substitutions applicable to modify  $\text{Sol}(\text{pb})$ .
10. *Solution Test and Validation.* The user selects the learned variation  $\Delta = \{\text{oil}^-, \text{peanut\_oil}^-, \text{olive\_oil}^+\}$  from the result set.  $\Delta$  is interpreted as the substitution  $\Sigma = \text{peanut\_oil} \rightsquigarrow \text{olive\_oil}$ , which suggests to replace peanut oil by olive oil in the retrieved recipe. The adaptation path  $\text{AP} = \Sigma \circ \Sigma_2$  is computed and the repaired solution  $\widetilde{\text{Sol}}(\text{tgt})$  is presented to the user for validation. The user is satisfied with the corrected solution  $\widetilde{\text{Sol}}(\text{tgt})$ , so the reformulation  $(\emptyset \rightsquigarrow \neg \text{peanut\_oil}, \text{peanut\_oil} \rightsquigarrow \text{olive\_oil})$  is added to the adaptation knowledge base  $\text{AKB}$ .

## 6 Discussion and Related Work

AK acquisition is a difficult task that is recognized to be a major bottleneck for CBR system designers due to the high knowledge-engineering costs it generates. To overcome these knowledge-engineering costs, a few approaches (e.g., [59,11]) have applied

machine learning techniques to learn AK offline from differences between cases of the case base. In [11], a set of pairs of source cases is selected from the case base and each selected pair of source cases is considered as a specific adaptation rule. The featural differences between problems constitute the antecedent part of the rule and the featural differences between solutions constitute the consequent part. Michalski's closing interval rule algorithm is then applied to generalize adaptation rule antecedents. In [9], adaptation knowledge takes the form of a set of adaptation cases. Each adaptation case associates an adaptation action to a representation of the differences between the two source problems. Machine learning algorithms like C4.5 or RISE are applied to learn generalized adaptation knowledge from these adaptation cases in order to improve the system's case-based adaptation procedure.

When applying machine learning techniques to learn adaptation knowledge from differences between cases, one main challenge concerns the choice of the training set: which cases are worth comparing? Arguing that (1) the size of the training set should be reduced to minimize the cost of the adaptation rule generation process and that (2) the source cases that are worth comparing should be the ones that are more similar, only the pairs of source cases that were judged to be similar according to a given similarity measure are selected in [9] and [11]. However, committing to a particular similarity measure might be somewhat arbitrary. Therefore, in [5], the authors decided to include in the training set all the pairs of distinct source cases of the case base. This paper introduces a third approach: the choice of the training set is determined interactively and according to the problem-solving context, taking advantage of the fact that the AK discovery process is triggered on-line. This approach appears to be very promising since the learning algorithm can be parametrized in order to learn only the knowledge that is needed to solve the target problem.

The examples presented above also show that knowledge discovery techniques allow to come up with more complex adaptation strategies than the simple one-to-one ingredient substitutions generated from the ontology  $\mathcal{O}$ . In particular, these techniques can help identifying interactions between the different ingredients that appear in the recipes (like e.g., that cocoa contains less fat than baking chocolate, so oil should be removed) as well as co-occurrences of ingredients (like say, that cinnamon is well-suited with apples). Besides, adaptation knowledge is learned at different levels of generality, so the user can be guided into gradual solution refinements.

Several CBR systems make use of interactive and/or opportunistic knowledge acquisition approaches to improve their learning capabilities. For example, in Creek, an approach that combines case-based and model-based methods, general knowledge is acquired through interactions with the user [1]. This knowledge acquisition process is provided in addition to the traditional case acquisition and allows the system to acquire knowledge that cannot be captured through cases only. In the Dial system, adaptation knowledge is acquired in the form of adaptation cases: when a case has to be adapted, the adaptation process is memorized in the form of a case and can be reused to adapt another case. Hence, adaptation knowledge is acquired through a CBR process inside the main CBR cycle. It must be remarked that adaptation cases can either be built automatically by adaptation of previous adaptation cases or manually by a user who interactively builds the adaptation case in response to a problem by selecting the



appropriates operations to perform [12]. Hence, knowledge acquisition in Dial appears to be both interactive and opportunistic. Chef is obviously related to the work described here [10]. Chef is a case-based planner in the cooking domain, its task is to build recipes on the basis of a user's request. The input of the system is a set of goals (tastes, textures, ingredients, types of dishes) and the output is a plan for a single recipe that satisfies all the goals. To solve this task, Chef is able to build new plans from old ones stored in memory. The system is provided with the ability to choose plans on the basis of the problems that they solve as well as the goals they satisfy, but it is also able to predict problems and to modify plans to avoid failures (plans are indexed in memory by the problems they avoid). Hence, Chef learns by providing causal explanations of failures thus marking elements as "predictive" of failures. In other words, the acquired knowledge allows the system to avoid identical failures to occur again. In our approach, we propose to go one step further by using failure to acquire knowledge that can be more widely used.

## 7 Conclusion and Future Work

In this paper, a novel approach for adaptation knowledge acquisition is presented in which the knowledge learned at problem-solving time by knowledge discovery techniques is directly reused for problem-solving. An application is proposed in the context of the cooking CBR system TAAABLE and the feasibility of the approach is demonstrated on some use cases. Future work will include developing a graphical user interface and doing more extensive testing. Opportunistic and interactive knowledge discovery in TAAABLE implies that the user plays the role of the domain expert, which raises several issues. For example, how to be sure that the knowledge expressed by a particular user is valuable? How to ensure that the adaptation knowledge base will remain consistent with time? Besides, TAAABLE is meant to be multi-user, so if the system's knowledge evolves with experience, some synchronization problems might occur. Therefore, the envisioned multi-user, ever-learning TAAABLE system needs to be thought of as a collaborative tool in which knowledge acquired by some users can be revised by others.

## References

1. Aamodt, A.: Knowledge-Intensive Case-Based Reasoning in Creek. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 1–15. Springer, Heidelberg (2004)
2. Badra, F., Lieber, J.: Representing Case Variations for Learning General and Specific Adaptation Rules. In: Cesta, A., Fakotakis, N. (eds.) Proceedings of the Fourth Starting AI Researcher's Symposium (STAIRS 2008), pp. 1–11 (2008)
3. Badra, F., Bendaoud, R., Bentebibel, R., Champin, P.-A., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A., Toussaint, Y.: Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In: Schaaf, M. (ed.) Computer Cooking Contest - Workshop at European Conference on Case-Based Reasoning (ECCBR 2008), pp. 219–228 (2008)
4. Schaaf, M. (ed.) ECCBR Workshops, ECCBR 2008, The 9th European Conference on Case-Based Reasoning, Workshop Proceedings (2008)

5. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case Base Mining for Adaptation Knowledge Acquisition. In: Proceedings of the International Conference on Artificial Intelligence, IJCAI 2007, pp. 750–756 (2007)
6. de Mántaras, R.L., Plaza, E.: Case-Based Reasoning: An Overview. *AI Communications* 10(1), 21–29 (1997)
7. Cordier, A.: Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning, Phd Thesis, Université Lyon 1 (2008)
8. Cordier, A., Fuchs, B., Lana de Carvalho, L., Lieber, J., Mille, A.: Opportunistic Acquisition of Adaptation Knowledge and Cases - The IakA Approach. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 150–164. Springer, Heidelberg (2008)
9. Craw, S., Wiratunga, N., Rowe, R.: Learning Adaptation Knowledge to Improve Case-Based Reasoning. *Artificial Intelligence* 170(16-17), 1175–1192 (2006)
10. Hammond, K.: CHEF: A model of case-based planning. In: Proceedings of the 5th National Conference on Artificial Intelligence, pp. 267–271. AAAI Press, Menlo Park (1986)
11. Hanney, K., Keane, M.T.: The Adaptation Knowledge Bottleneck: How to Unblock it By Learning From Cases. In: Proceedings of the 2nd International Conference on CBR, pp. 359–370 (1997)
12. Leake, D., Kinley, A., Wilson, D.: Acquiring Case Adaptation Knowledge: A Hybrid Approach. In: Proc. of the 13th National Conference on Artificial Intelligence, pp. 684–689 (1996)
13. Melis, E., Lieber, J., Napoli, A.: Reformulation in Case-Based Reasoning. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 172–183. Springer, Heidelberg (1998)

# Improving Reinforcement Learning by Using Case Based Heuristics

Reinaldo A.C. Bianchi<sup>1,2</sup>, Raquel Ros<sup>2</sup>, and Ramon Lopez de Mantaras<sup>2</sup>

<sup>1</sup> Centro Universitário da FEI, São Bernardo do Campo, Brazil  
rbianchi@fei.edu.br

<sup>2</sup> Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain  
{ros,mantaras}@iiia.csic.es

**Abstract.** This work presents a new approach that allows the use of cases in a case base as heuristics to speed up Reinforcement Learning algorithms, combining Case Based Reasoning (CBR) and Reinforcement Learning (RL) techniques. This approach, called Case Based Heuristically Accelerated Reinforcement Learning (CB-HARL), builds upon an emerging technique, the Heuristic Accelerated Reinforcement Learning (HARL), in which RL methods are accelerated by making use of heuristic information. CB-HARL is a subset of RL that makes use of a heuristic function derived from a case base, in a Case Based Reasoning manner. An algorithm that incorporates CBR techniques into the Heuristically Accelerated Q-Learning is also proposed. Empirical evaluations were conducted in a simulator for the RoboCup Four-Legged Soccer Competition, and results obtained shows that using CB-HARL, the agents learn faster than using either RL or HARL methods.

## 1 Introduction

Case Based Reasoning (1; 2) techniques have been shown to be useful in a multitude of domains, with widespread applications ranging from the optimization of autoclave loading (3), the diagnosis and treatment of many medical problems (4), to the synthesis of high quality expressive music (5).

Reinforcement Learning (RL) is also a very successful Artificial Intelligence sub-area. RL algorithms are very useful for solving a wide variety problems when their models are not available a priori, since many of them are known to have guarantees of convergence to equilibrium (6; 7). Unfortunately, the convergence of a RL algorithm may only be achieved after an extensive exploration of the state-action space, which is usually very time consuming.

One way to speed up the convergence of RL algorithms is making use of a conveniently chosen heuristic function, which can be used for selecting the appropriate actions to perform in order to guide exploration during the learning process. Several Heuristically Accelerated Reinforcement Learning (HARL) methods that makes use of a heuristic function have been recently proposed (8; 9). These techniques are very attractive: as RL, they are based on firm theoretical foundations. As the heuristic function is used only in the choice of the action to be taken, many of the conclusions obtained

for RL remain valid for HARL algorithms, such as the guarantee of convergence to equilibrium in the limit and the definition of an upper bound for the error.

Although several methods have been successfully applied for defining the heuristic function, a very interesting option had not been explored yet: the reuse of previously learned policies, using a Case Based Reasoning approach to define an heuristic function. This paper investigates the combination of Case Based Reasoning (CBR) and Heuristically Accelerated Reinforcement Learning (HARL) techniques, with the goal of speeding up RL algorithms by using previous domain knowledge, stored as a case base. To do so, we propose a new algorithm, the Case Based Heuristically Accelerated Q–Learning (CB-HAQL), which incorporates Case Based Reasoning techniques into an existing HARL algorithm, the Heuristically Accelerated Q–Learning (HAQL).

The application domain of this paper is that of the RoboCup Standard Platform League, Four-Legged Soccer Competition (10), where teams consisting of four Sony AIBO robots operating fully autonomously and communicating through a wireless network compete in a 6 x 4 m field. This domain is one of many RoboCup challenges, which has been proven to be an important domain for research, and where RL techniques have been widely used. Nevertheless, the technique proposed in this work is domain independent.

The paper is organized as follows: Section 2 briefly reviews the Reinforcement Learning problem; Section 3 describes the HARL approach and the HAQL algorithm, while section 4 describes Case Based Reasoning. Section 5 shows how to incorporate CBR techniques into HARL algorithms, in a modified formulation of the HAQL algorithm. Section 6 describes the robotic soccer domain used in the experiments, presents the experiments performed, and shows the results obtained. Finally, conclusions are presented in Section 7.

## 2 Reinforcement Learning and the Q–Learning Algorithm

Reinforcement Learning (RL) algorithms have been applied successfully to the on-line learning of optimal control policies in Markov Decision Processes (MDPs). In RL, this policy is learned through trial-and-error interactions of the agent with its environment: on each interaction step the agent senses the current state  $s$  of the environment, chooses an action  $a$  to perform, executes this action, altering the state  $s$  of the environment, and receives a scalar reinforcement signal  $r$  (a reward or penalty).

The RL problem can be formulated as a discrete time, finite state, finite action Markov Decision Process (MDP). The learning environment can be modeled by a 4-tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ , where:

- $\mathcal{S}$ : is a finite set of states.
- $\mathcal{A}$ : is a finite set of actions that the agent can perform.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ : is a state transition function, where  $\Pi(\mathcal{S})$  is a probability distribution over  $\mathcal{S}$ .  $T(s, a, s')$  represents the probability of moving from state  $s$  to  $s'$  by performing action  $a$ .
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathfrak{R}$ : is a scalar reward function.

The goal of the agent in a RL problem is to learn an optimal policy  $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$  that maps the current state  $s$  into the most desirable action  $a$  to be performed in  $s$ .

**Table 1.** The Q–Learning algorithm

---

Initialize  $\hat{Q}_t(s, a)$  arbitrarily.  
Repeat (for each episode):  
  Initialize  $s$  randomly.  
  Repeat (for each step):  
    Select an action  $a$  using equation [2](#).  
    Execute the action  $a$ , observe  $r(s, a), s'$ .  
    Update the values of  $Q(s, a)$  according to equation [1](#).  
     $s \leftarrow s'$ .  
  Until  $s$  is terminal.  
Until some stopping criterion is reached.

---

One strategy to learn the optimal policy  $\pi^*$  is to allow the agent to learn the evaluation function  $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ . Each action value  $Q(s, a)$  represents the expected cost incurred by the agent when taking action  $a$  at state  $s$  and following an optimal policy thereafter.

The Q–learning algorithm ([11](#)) is a well-know RL technique that uses a strategy to learn an optimal policy  $\pi^*$  via learning of the action values. It iteratively approximates  $Q$ , provided the system can be modeled as an MDP, the reinforcement function is bounded, and actions are chosen so that every state-action pair is visited an infinite number of times (the complete algorithm is presented in Table [1](#)). The  $Q$  learning update rule is:

$$\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha \left[ r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a) \right], \quad (1)$$

where  $s$  is the current state;  $a$  is the action performed in  $s$ ;  $r$  is the reward received;  $s'$  is the new state;  $\gamma$  is the discount factor ( $0 \leq \gamma < 1$ ); and  $\alpha$  is the learning rate. To select an action to be executed, the Q–Learning algorithm usually considers an  $\epsilon - Greedy$  strategy:

$$\pi(s) = \begin{cases} \arg \max_a \hat{Q}(s, a) & \text{if } q \leq p, \\ a_{random} & \text{otherwise} \end{cases} \quad (2)$$

where:

- $q$  is a random value uniformly distributed over  $[0, 1]$  and  $p$  ( $0 \leq p \leq 1$ ) is a parameter that defines the exploration/exploitation tradeoff: the larger  $p$ , the smaller is the probability of executing a random exploratory action.
- $a_{random}$  is an action randomly chosen among those available in state  $s$ .

In RL, learning is carried out online, through trial-and-error interactions of the agent with the environment. Unfortunately, convergence of any RL algorithm may only be achieved after extensive exploration of the state-action space. In the next section we show one way to speed up the convergence of RL algorithms, by making use of a heuristic function in a manner similar to the use of heuristics in informed search algorithms.

### 3 Heuristic Accelerated Reinforcement Learning and the HAQL Algorithm

Formally, a Heuristically Accelerated Reinforcement Learning (HARL) algorithm (8) is a way to solve a MDP problem with explicit use of a heuristic function  $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \mathfrak{R}$  for influencing the choice of actions by the learning agent.  $H(s, a)$  defines the heuristic that indicates the importance of performing the action  $a$  when visiting state  $s$ . The heuristic function is strongly associated with the policy: every heuristic indicates that an action must be taken regardless of others.

The heuristic function is an action policy modifier, which does not interfere with the standard bootstrap-like update mechanism of RL algorithms. A possible strategy for action choice is an  $\epsilon$ -greedy mechanism where a heuristic mechanism formalized as a function  $H(s, a)$  is considered, thus:

$$\pi(s) = \begin{cases} \arg \max_a [F(s, a) \bowtie \xi H(s, a)^\beta] & \text{if } q \leq p, \\ a_{\text{random}} & \text{otherwise} \end{cases} \quad (3)$$

where:

- $\mathcal{F} : \mathcal{S} \times \mathcal{A} \rightarrow \mathfrak{R}$  is an estimate of a value function that defines the expected cumulative reward. If  $F(s, a) \equiv \hat{Q}(s, a)$  we have an algorithm similar to standard *Q-Learning*.
- $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \mathfrak{R}$  is the heuristic function that plays a role in the action choice.  $H(s, a)$  defines the importance of executing action  $a$  in state  $s$ .
- $\bowtie$  is a function that operates on real numbers and produces a value from an ordered set which supports a maximization operation.
- $\xi$  and  $\beta$  are design parameters that control the influence of the heuristic function.
- $q$  is a parameter that defines the exploration/exploitation tradeoff.
- $a_{\text{random}}$  is an action randomly chosen among those available in state  $s$ .

The first HARL algorithm proposed was the Heuristically Accelerated Q-Learning (HAQL) (8), as an extension of the Q-Learning algorithm. The only difference between them is that in the HAQL makes use of an heuristic function in the action choice rule defined in Equation (3), where  $\mathcal{F} = Q$ , the  $\bowtie$  operator is the sum, and  $\beta = 1$ :

$$\pi(s) = \begin{cases} \arg \max_a [\hat{Q}(s, a) + \xi H(s, a)] & \text{if } q \leq p, \\ a_{\text{random}} & \text{otherwise,} \end{cases} \quad (4)$$

where all variables are defined as in Equation (3).

As a general rule, the value of  $H(s, a)$  used in HAQL should be higher than the variation among the  $\hat{Q}(s, a)$  values for the same  $s \in \mathcal{S}$ , in such a way that it can influence the choice of actions, and it should be as low as possible in order to minimize the error. It can be defined as:

$$H(s, a) = \begin{cases} \max_i \hat{Q}(s, i) - \hat{Q}(s, a) + \eta & \text{if } a = \pi^H(s), \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

**Table 2.** The HAQL algorithm

---

Initialize  $\hat{Q}_t(s, a)$  and  $H_t(s, a)$  arbitrarily.

Repeat (for each episode):

  Initialize  $s$ .

  Repeat (for each step):

    Update the values of  $H_t(s, a)$  as desired.

    Select an action  $a$  using equation 4

    Execute the action  $a$ , observe  $r(s, a)$ ,  $s'$ .

    Update the values of  $Q(s, a)$  according to equation 11

$s \leftarrow s'$ .

  Until  $s$  is terminal.

Until some stopping criterion is reached.

---

where  $\eta$  is a small real value (usually 1) and  $\pi^H(s)$  is the action suggested by the heuristic policy.

Convergence of the HAQL algorithm was presented by Bianchi, Ribeiro and Costa (8), together with the definition of an upper bound for the error in the estimation of  $Q$ . The complete HAQL algorithm is presented in Table 2. The same authors investigated the use of HARL in multiagent domain, proposing a multiagent HARL algorithm – the Heuristically Accelerated Minimax-Q (9) – and testing it in a simplified simulator for the robot soccer domain.

Despite the fact that RL is a method that has been traditionally applied in Robotic Soccer domains, only recently have HARL methods been used in this domain. Bianchi, Ribeiro and Costa (9) investigated the use of a multiagent HARL algorithm in a simplified simulator for the robot soccer domain and Celiberto, Ribeiro, Costa and Bianchi (12) studied the use of the HARL algorithms to speed up learning in the RoboCup 2D Simulation domain.

## 4 Case Based Reasoning

Case based reasoning (CBR) (1; 2) uses knowledge of previous situations (cases) to solve new problems, by finding a similar past case and reusing it in the new problem situation. In the CBR approach, a case usually describes a problem and its solution, i.e., the state of the world in a defined moment and the sequence of actions to perform to solve that problem.

According to López de Mántaras *et al* (2), solving a problem by CBR involves “obtaining a problem description, measuring the similarity of the current problem to previous problems stored in a case base with their known solutions, retrieving one or more similar cases, and attempting to reuse the solution of the retrieved case(s), possibly after adapting it to account for differences in problem descriptions”. Other steps that are usually found in CBR systems are the evaluation of the proposed solution, the revision of the solution, if required in light of its evaluation, and the retention (learning) of a new case, if the system has learned to solve a new problem.

The case definition used in this work is the one proposed by Ros (13; 14; 15; 16), which is composed of three parts: the problem description ( $P$ ), the solution description ( $A$ ) and the case scope ( $K$ ), and is formally described as a 3-tuple:

$$case = (P, A, K).$$

The problem description  $P$  corresponds to the situation in which the case can be used. For example, for a simple robotic soccer problem, the description of a case can include the robot position, the ball's position and the positions of the other robots in the game. For a game with  $n$  robots,  $P$  can be:

$$P = \{x_B, y_B, x_{R_0}, y_{R_0}, \dots, x_{R_n}, y_{R_n}\}.$$

The solution description is composed by the sequence of actions that each robot must perform to solve the problem, and can be defined as:

$$A = \{R_0 : [a_{0_1}, a_{0_2}, \dots, a_{0_{p_0}}], \dots, R_n : [a_{n_1}, a_{n_2}, \dots, a_{n_{p_n}}]\},$$

where  $n$  is the number of robots in the team,  $a_{0_i}$  is an individual or joint action that robot  $R_i$  must perform and  $p_i$  corresponds the number of actions the robot  $R_i$  performs.

The case scope defines the applicability boundaries of the cases, to be used in the retrieval step. For example, Ros (16) define it as "the regions of the field within which the ball and the opponents should be positioned in order to retrieve that case". In the case of a simple robot soccer problem,  $K$  can be represented as ellipsoids centered on the ball's and opponents' positions indicated in the problem description. It can be defined as:

$$K = \{(\tau_B^x, \tau_B^y), (\tau_{R_0}^x, \tau_{R_0}^y) \dots, (\tau_{R_n}^x, \tau_{R_n}^y)\},$$

where  $\tau_B^x, \tau_B^y$  corresponds to the  $x$  and  $y$  radius of the ellipsoid region around the ball and  $(\tau_{R_0}^x, \tau_{R_0}^y) \dots, (\tau_{R_n}^x, \tau_{R_n}^y)$  the radius of the regions around the  $n$  robots in the game (teammates and opponents).

Case retrieval is in general driven by a similarity measure between the new problem and the solved problems in the case base. In this work we use the case retrieval method proposed by Ros (16), where the similarity along three important aspects are evaluated:

- the similarity between the problem and the case;
- the cost of adapting the problem to the case, and;
- the applicability of the solution of the case.

The similarity function indicates how similar a problem and a case are. In most cases, the function is defined by the distance between the ball and the robots in the problem and in the case.

$$Sim(p, c) = dist(B^c, B^p) + \sum_{i=0}^n dist(R_i^c, R_i^p),$$

where  $B^c$  is the position of the ball in the case and  $B^p$  its position in the problem,  $R_i^c$  the position of the Robot  $i$  in the case and  $R_i^p$  its position in the problem, and  $dist(a, b)$  is the Gaussian distance between object  $a$  and  $b$ . This distance is computed as follows:



$$dist(a, b) = exp \left( - \left[ \left( \frac{a_x - b_x}{\tau^x} \right)^2 + \left( \frac{a_y - b_y}{\tau^y} \right)^2 \right] \right),$$

where  $\tau^x, \tau^y$  are the radius of the scope around the object (ball and robots positions). The Gaussian distance is used because the larger the distance between two points, the lower the similarity between them. Also, the  $\tau^x, \tau^y$  parameters are used as a threshold that defines a maximum distance allowed for two points to have some degree of similarity: if the distance is greater than a limit,  $Sim(a, b) = 0$ .

The cost of adapting the problem to the case is computed as a function of the distances between the positions of the team robots in the problem and the positions specified in the case. The adaptation cost is defined as:

$$cost(p, c) = \sum_{i=1}^n dist(r_i, adaptPos_i)$$

where  $n$  is the number of robots that take part of the case solution,  $dist$  is the Euclidian distance,  $r_i$  is the current position of robot  $i$  and  $adaptPos_i$  the adapted position for robot  $i$ .

The applicability of the solution of the case depends on the position of the opponents, and combine two functions: the free path function, which considers if the trajectory of the ball indicated in the case is free of opponents, in order for the evaluated case to be applicable and the opponent similarity, which computes if the opponents represent a significant threat for the robots to fulfill the task, such as an opponent blocking the ball or an opponent located near enough to get the ball first. These functions and the complete case retrieval algorithm are described in detail in Ros (16).

In recent years, CBR has been used by several researchers in the Robotic Soccer domain. To mention a few, Lin, Liu and Chen (17) presented a hybrid architecture for soccer players where the deliberative layer corresponds to a CBR system, Ahmadi *et al* (18) presented a two-layered CBR system for prediction for the coach, and Karol *et al* (19) presented high level planning strategies including a CBR system. Finally, the works of Ros (13) presents the most ample use of CBR techniques in the Robotic Soccer domain, proposing the use of CBR techniques to handle retrieval, reuse and acquisition of a case base for the action selection problem of a team for the Four-Legged robots.

## 5 Combining Case Based Reasoning and Reinforcement Learning

In order to give HARK algorithms the capability of reusing previous knowledge from a domain, we propose a new algorithm, the Case Based HAQL, which extends the HAQL algorithm with the abilities to retrieve a case stored in a base, adapt it to the current situation, and build a heuristic function that corresponds to the case.

As the problem description  $P$  corresponds to one defined state of the set of states  $S$  in an MDP, an algorithm that uses the RL loop can be implemented. Inside this loop, before the action selection, we added steps to compute the similarity of the cases in the base with the current state and the cost of adaptation of these cases. A case is retrieved if the similarity is above a certain threshold, and adaptation cost is low. After a case is retrieved, an heuristic is computed using Equation 5 with the sequence of actions

**Table 3.** The CB-HAQL algorithm

---

```

Initialize  $\hat{Q}_t(s, a)$  and  $H_t(s, a)$  arbitrarily.
Repeat (for each episode):
  Initialize  $s$ .
  Repeat (for each step):
    Compute similarity and cost.
    If there is a case that can be reused:
      Retrieve and Adapt if necessary.
      Compute  $H_t(s, a)$  using Equation 5 with the
        actions suggested by the case selected.
    Select an action  $a$  using equation 4.
    Execute the action  $a$ , observe  $r(s, a)$ ,  $s'$ .
    Update the values of  $Q(s, a)$  according to equation 1.
     $s \leftarrow s'$ .
  Until  $s$  is terminal.
Until some stopping criterion is reached.

```

---

suggested by the case selected, and this heuristic is used for a certain amount of time, equal to the number of actions of the retrieved case. After that time, a new case can be retrieved. The complete CB-HAQL algorithm is presented in Table 3.

Although this is the first paper to combine CBR with RL by the use of an explicit heuristic function, this is not the first work on combining the CBR and RL fields. Drummond (20) was probably the first to use CBR to speed up RL, proposing to accelerate RL by transferring parts of previously learned solutions to a new problem, exploiting the results of prior learning to speed up the process. The system identifies subtasks on the basis of stable features that arise in the multi-dimensional value function due to the interaction of the learning agent with the world during the learning process. As the agent learns, it can access a case base that contains clipped parts of previous learned value functions that solves individual problems, speeding up the convergence time.

Several other authors have been studying the use of RL together with CBR and the relation between them. Sharma *et al* (21) makes use of CBR as a function approximator for RL, and RL as revision algorithm for CBR in a hybrid architecture system; Gabel and Riedmiller (22) also makes use of CBR in the task of approximating a function over high-dimensional, continuous spaces; Juell and Paulson (23) exploit the use of RL to learn similarity metrics in response to feedback from the environment; Auslander *et al* (24) uses CBR to adapt quickly an RL agent to changing conditions of the environment by the use of previously stored policies and Li, Zonghai and Feng (25) proposes an algorithm that makes use of knowledge acquired by reinforcement learning to construct and extend a case base.

Finally, works on Transfer Learning have also combined CBR and RL. For example, van Helsing and Goel (26) describes a technique for abstracting reusable cases from RL, enabling the transfer of acquired knowledge to other instances of the same problem.

Our approach differs from all previous research combining CBR and RL because of the heuristic use of the retrieved case: as the case is used only as a heuristic, if the case

base contains a case that can be used in one situation, there will be a speed up in the convergence time. But if the case base does not contain any useful case – or even if it contains cases that implement wrong solutions to the problem, the agent will learn the optimal solution anyway, by using the RL component of the algorithm.

## 6 Experiments in the Robotic Soccer Domain

Empirical evaluations of the CB-HAQL approach were carried out in an extended version of the PuppySim 2 simulator, created by the CMDash team (27) and extended by Ros (16). This simulator represents the basic aspects of the RoboCup Standard Platform League, Four-Legged Soccer Competition (28), and is used to test the robot’s behavioral response under ideal environmental conditions: the robots’ perception is noiseless, but the outcome of the actions the robots perform have a certain degree of randomness.

Using this simulator experiments were performed using two attackers against a defender and a goalie. The attackers are two robots controlled by one of the algorithms to be evaluated: the Q-Learning, described in section 2, the HAQL, described in section 3 or the CB-HAQL, proposed in section 5 and we have also compared them to the results of the CBR system alone obtained by Ros (16). The opponents perform the same reactive behavior when playing against any of the evaluated approaches. The defender and the goalie have a home region which cannot go beyond. If the ball is within its home region, then the robot moves towards the ball and clears it. Otherwise, the robot remains in the boundary of its home region, facing the ball to maintain it in its point of view. Each trial begins with the attackers being positioned in the field in a random position, and the defender, the goalie and the ball in a fixed location (ball in the center, and defender and goalie in the center of their home region). Figure 1 shows the PuppySim 2

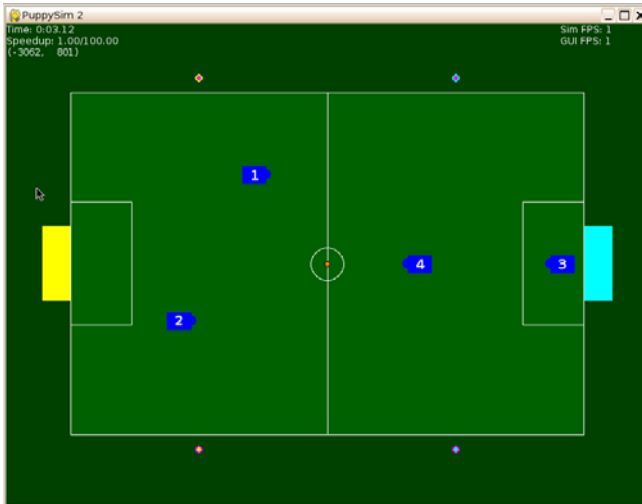
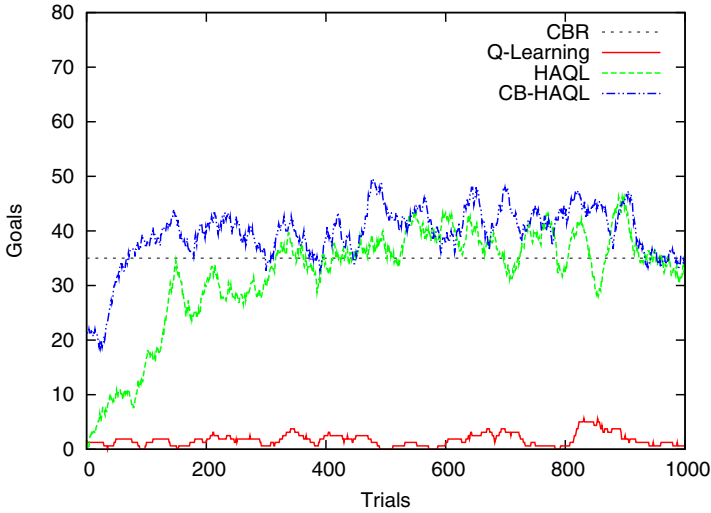


Fig. 1. The PuppySim2 users’ interface showing the robots at their initial positions



**Fig. 2.** Percentage of goals scored in each trial using the CBR (constant line at 35%), Q-learning, the HAQL and the CB-HAQL algorithms

users' interface with one starting configuration. A trial ends either when the attackers score a goal, the ball goes out of the field or the goalie touches it.

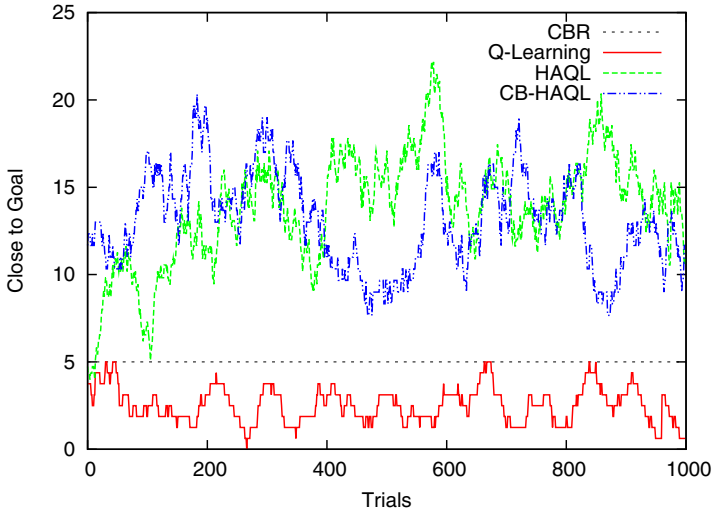
The heuristic used in the HAQL algorithm was defined using a simple rule: if holding the ball, go to the opponents goal, not taking into account the opponents positions, leaving the task of how to divert the opponent to the learning process. The heuristic used in the CB-HAQL is computed during the games, as described in section 5. The case base used for the experimentation is composed of 136 cases, which cover the most significant situations that can occur in the evaluation presented in this work. From this set, 34 cases are initially defined, while the remaining ones are automatically generated using spatial transformations exploiting the symmetries of the soccer field. The reward the agents receive are the same for all algorithms: the agent receives  $-100$  every time the ball go out of the field or the goalie touches it and  $+100$  a robot scores a goal.

In order to evaluate each trial we classify the possible outcomes as:

- goal : the ball enters the goal.
- close : the ball goes out of the field but passes near one of the goalposts. More precisely, at most 25cm to the left (right) of the left (right) goalpost.
- block : the goalie stops or kicks the ball.
- out : the ball goes out the field without being a goal or close to goal.

We also consider the “to-goal” balls, which correspond to balls that are either goals or close to goal. This measure indicates the degree of goal intention of the kicks. Thus, although the balls might not enter the goal, at least they were intended to do so.

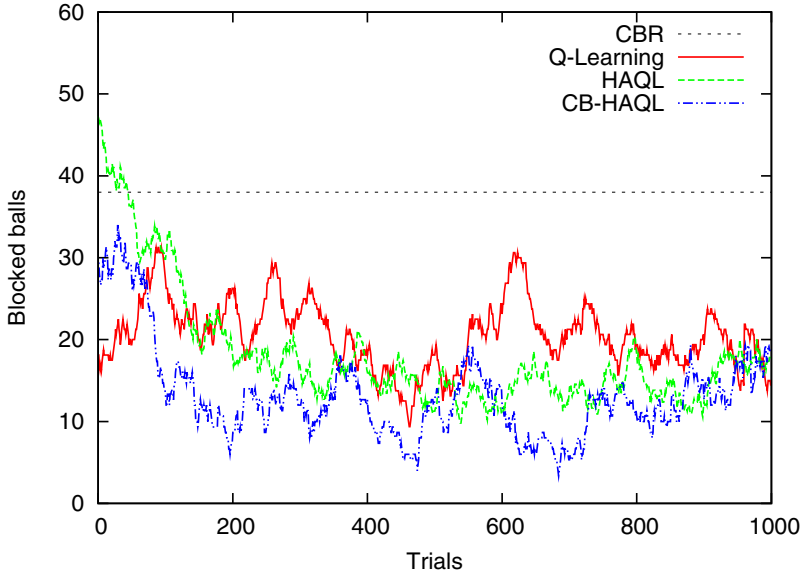
Twenty five training sessions were run for the three algorithms, with each session consisting of 1000 trials. Figures 2 to 5 shows the learning curves for all algorithms



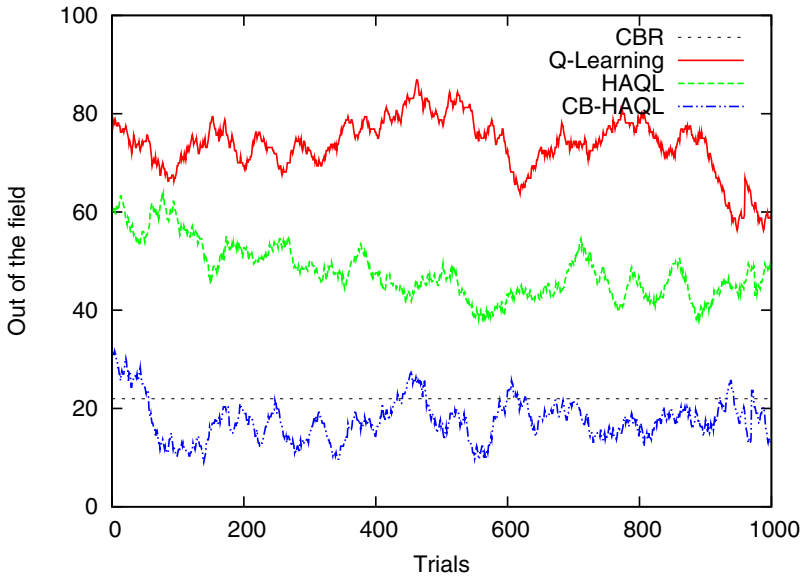
**Fig. 3.** Percentage of balls that passed close to the goals in each trial, for the CBR (constant line at 5%), Q-learning, the HAQL and the CB-HAQL algorithms

(the CBR alone and the three learning algorithms) and presents the percent of goals scored (Fig. 2) by the learning team, balls that passed close to the goals (Fig. 3), balls blocked by the defenders (Fig. 4) and balls that went out (Fig. 5) in each trial. It is possible to verify in Fig. 2 that at the beginning of the learning phase HAQL has worse performance than the CB-HAQL, and as the trials proceed, the performance of both algorithms become similar, as expected, since all the algorithms converge to equilibrium. The Q-learning is clearly the one with the worst performance, since it takes much more trials for it to start to learn even basic policies, as not to kick the ball out of the field. In this figure it can also be observed the performance of two agents using only the case base. Student's  $t$ -test was used to verify the hypothesis that the use of heuristics speeds up the learning process. Using the data from Fig. 2, the result is that the CB-HAQL is better (makes more goals) than HAQL and Q-Learning until the 300<sup>th</sup> trial, with a level of confidence greater than 5%. After this trial the results of the CB-HAQL and HAQL are comparable.

Finally, table 4 summarizes the ball classification outcome obtained (results in percentage) using the CBR approach and the three learning algorithms. The results for the CBR approach are the average of 500 trials, and the results for the Q-learning, HAQL and CB-HAQL are the average of 100 trials, using the Q-table that the three algorithms had at the end of the 1000<sup>th</sup> trial. As we can see the percentage of balls to goal with the CB-HAQL approach is higher compared to either the HAQL or the Q-Learning algorithms. Moreover, the percentage of balls out are lower when using CBR, indicating that the defender had less opportunities to take the ball and kick it out of the field, and that the agent performed less random exploration.



**Fig. 4.** Percentage of balls blocked by the defenders for the CBR (constant line at 38%), Q-learning, the HAQL and the CB-HAQL algorithms



**Fig. 5.** Percentage of balls that went out of the field in each trial, for the CBR (constant line at 22%), Q-learning, the HAQL and the CB-HAQL algorithms

**Table 4.** Ball outcome classification (results in percentage)

Approach	Goal	Close	To-Goal Goal + Close	Blocked	Out
CBR	35	5	40	38	22
Q-Learning	2	2	4	22	74
HAQL	16	4	20	20	60
CB-HAQL	40	7	47	36	17

The parameters used in the experiments were the same for all the algorithms:  $\alpha = 0,9$ , the exploration/ exploitation = 0.2,  $\gamma = 0.9$  and  $\eta = 1$  Values in the Q table were randomly initiated. The experiments were programmed in Python and executed in a MacBook Pro, with 4GB of RAM in a Mac OS X platform.

## 7 Conclusion

This work presented a new algorithm, called Case Based Heuristically Accelerated Q-Learning (CB-HAQL), which allows the use of a case base to define heuristics to speed up the well-known Reinforcement Learning algorithm Q-Learning. This approach builds upon an emerging technique, the Heuristic Accelerated Reinforcement Learning (HARL), in which RL methods are accelerated by making use of heuristic information.

The experimental results obtained showed that CB-HAQL attained better results than HAQL and Q-Learning for the domain of robotic soccer games. For example, the Q-Learning, after 1000 learning trials, still could not produce policies that scored goals on the opponent, while the HAQL was able to score some goals but significantly less than the CBR alone and the CB-HAQL. Another interesting finding is that the goals scored by the CB-HAQL after 1000 trials was even slightly higher than the number of goals scored by the CBR approach alone, indicating that the learning component of the CB-HAQL algorithm was able to improve the initial case base.

Another important finding of this work is that the CBR approach generated better results than the Q-learning algorithm, for the same experimental setup. Experiments executed until the 10.000<sup>th</sup> trial showed that the Q-Learning still had not converged, indicating the slow rate of learning of this algorithm, in this domain.

Finally, since Heuristic functions allow RL algorithms to solve problems where the convergence time is critical, as in many real time applications, in future work we plan to incorporate CBR in other well known RL algorithms, like SARSA, Q( $\lambda$ ), Minimax-Q, Minimax-Q( $\lambda$ ), and Nash-Q, and expand this framework to deal with General Sum Markov Games.

**Acknowledgement.** This work has been partially funded by the 2005-SGR-00093 grant of the Generalitat de Catalunya, the MID-CBR project TIN 2006-15140-C03-01, and FEDER funds. Reinaldo Bianchi acknowledge the support of the CNPq (Grant No. 201591/2007-3) and FAPESP (Grant No. 2009/01610-1).

## References

- [1] Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun.* 7(1), 39–59 (1994)
- [2] de Mántaras, R.L., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision and retention in case-based reasoning. *Knowl. Eng. Rev.* 20(3), 215–240 (2005)
- [3] Hennessy, D., Hinkle, D.: Applying case-based reasoning to autoclave loading. *IEEE Expert: Intelligent Systems and Their Applications* 7(5), 21–26 (1992)
- [4] Althoff, K.D., Bergmann, R., Wess, S., Manago, M., Auriol, E., Larichev, O.I., Bolotov, A., Zhuravlev, Y.I., Gurov, S.I.: Case-based reasoning for medical decision support tasks: The inreca approach. In: *Artificial Intelligence in Medicine*, January 1998, pp. 25–41 (1998)
- [5] López de Mántaras, R., Cunningham, P., Perner, P.: Emergent case-based reasoning applications. *Knowl. Eng. Rev.* 20(3), 325–328 (2005)
- [6] Szepesvári, C., Littman, M.L.: Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms. Technical report, Brown University, CS-96-11 (1996)
- [7] Littman, M.L., Szepesvári, C.: A generalized reinforcement learning model: convergence and applications. In: *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*, pp. 310–318 (1996)
- [8] Bianchi, R.A.C., Ribeiro, C.H.C., Costa, A.H.R.: Accelerating autonomous learning by using heuristic selection of actions. *Journal of Heuristics* 14(2), 135–168 (2008)
- [9] Bianchi, R.A.C., Ribeiro, C.H.C., Costa, A.H.R.: Heuristic selection of actions in multi-agent reinforcement learning. In: Veloso, M.M. (ed.) *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, January 6-12, pp. 690–695 (2007)
- [10] RoboCup Technical Committee: Standard platform league homepage (2009), <http://www.tzi.de/spl>
- [11] Watkins, C.J.C.H.: Learning from Delayed Rewards. PhD thesis, University of Cambridge (1989)
- [12] Celiberto, L.A., Ribeiro, C.H.C., Costa, A.H.R., Bianchi, R.A.C.: Heuristic reinforcement learning applied to robocup simulation agents. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) *RoboCup 2007: Robot Soccer World Cup XI*. LNCS, vol. 5001, pp. 220–227. Springer, Heidelberg (2008)
- [13] Ros, R., Arcos, J.L., de Mantaras, R.L., Veloso, M.: A case-based approach for coordinated action selection in robot soccer. *Artificial Intelligence* 173(9-10), 1014–1039 (2009)
- [14] Ros, R., de Mántaras, R.L., Arcos, J.L., Veloso, M.: Team playing behavior in robot soccer: A case-based approach. In: Weber, R.O., Richter, M.M. (eds.) *ICCBR 2007*. LNCS (LNAI), vol. 4626, pp. 46–60. Springer, Heidelberg (2007)
- [15] Ros, R., Arcos, J.L.: Acquiring a robust case base for the robot soccer domain. In: Veloso, M. (ed.) *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 1029–1034. AAAI Press, Menlo Park (2007)
- [16] Ros, R.: Action Selection in Cooperative Robot Soccer using Case-Based Reasoning. PhD thesis, Universitat Autònoma de Barcelona, Barcelona (2008)
- [17] Lin, Y., Liu, A., Chen, K.: A hybrid architecture of case-based reasoning and fuzzy behavioral control applied to robot soccer. In: *Workshop on Artificial Intelligence, International Computer Symposium (ICS 2002)*, Hualien, Taiwan, National Dong Hwa University, National Dong Hwa University (2002)
- [18] Ahmadi, M., Lamjiri, A.K., Nevisi, M.M., Habibi, J., Badie, K.: Using a two-layered case-based reasoning for prediction in soccer coach. In: Arabnia, H.R., Kozerenko, E.B. (eds.) *MLMTA*, pp. 181–185. CSREA Press (2003)



- [19] Karol, A., Nebel, B., Stanton, C., Williams, M.A.: Case based game play in the robocup four-legged league part i the theoretical model. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS, vol. 3020, pp. 739–747. Springer, Heidelberg (2004)
- [20] Drummond, C.: Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of Artificial Intelligence Research* 16, 59–104 (2002)
- [21] Sharma, M., Holmes, M., Santamaría, J.C., Irani, A., Isbell Jr., C.L., Ram, A.: Transfer learning in real-time strategy games using hybrid cbr/rl. In: Veloso, M.M. (ed.) IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, pp. 1041–1046 (2007)
- [22] Gabel, T., Riedmiller, M.A.: CBR for state value function approximation in reinforcement learning. In: Muñoz-Avila, H., Ricci, F. (eds.) ICCBR 2005. LNCS, vol. 3620, pp. 206–221. Springer, Heidelberg (2005)
- [23] Juell, P., Paulson, P.: Using reinforcement learning for similarity assessment in case-based systems. *IEEE Intelligent Systems* 18(4), 60–67 (2003)
- [24] Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 59–73. Springer, Heidelberg (2008)
- [25] Li, Y., Zonghai, C., Feng, C.: A case-based reinforcement learning for probe robot path planning. In: 4th World Congress on Intelligent Control and Automation, Shanghai, China, pp. 1161–1165 (2002)
- [26] von Hessling, A., Goel, A.K.: Abstracting reusable cases from reinforcement learning. In: Brüninghaus, S. (ed.) ICCBR Workshops, pp. 227–236 (2005)
- [27] Veloso, M., Rybski, P.E., Chernova, S., McMillen, C., Fasola, J., von Hundelshausen, F., Vail, D., Trevor, A., Hauert, S., Ros, R.: Cmdash 2005: Team report. Technical report, School of Computer Science, Carnegie Mellon University (2005)
- [28] RoboCup Technical Committee: RoboCup Four-Legged League Rule Book (2008)

# Dimensions of Case-Based Reasoner Quality Management

Annett Bierer<sup>1</sup> and Marcus Hofmann<sup>2</sup>

Chemnitz University of Technology

Faculty for economics and business administration

<sup>1</sup>Chair of management accounting and controlling

<sup>2</sup>Chair of business process management and knowledge management

Thüringer Weg 7, D-09126 Chemnitz, Germany

Tel.: +49(0)371/531-33975; Fax: +49 (0) 371/531-26529

{annett.bierer,marcus.hofmann}@wirtschaft.tu-chemnitz.de

**Abstract.** CBR systems are increasingly applied in practice. There, the system's ability to provide users with the information they need depends on the quality of the system's knowledge base and its long-term assurance and continuous improvement. Currently, in CBR research quality is rather an abstract term and maintenance takes a center stage. But in practice, there is a need for a broader and more detailed focus on quality aspects. Hence, the paper extends the maintenance view to a quality-oriented view. A framework is presented containing relevant aspects of a CBR system's quality management (CBRQM). It will be shown that maintenance approaches are not independent from a CBRQM.

**Keywords:** case-based reasoner quality management, case-based reasoner maintenance, CBRQM framework.

## 1 Introduction

Today, more and more CBR systems are applied in practice, e.g., in medicine, credit assessment, aerospace. The growing use of large-scaled and long-term CBR systems has brought with it an increased demand for systematic maintenance, especially of the knowledge items in the system's knowledge base. This is because the quality of the knowledge items, i.e. its ability to provide appropriate information and solution suggestions to the users in a given application domain, is vitally important for the acceptance of the CBR system in a company.

In the past, maintenance approaches and methods (maintenance algorithms, concepts to guide the maintenance process) for preserving, restoring or improving the quality of the knowledge items have been developed. Up to now, it is often disregarded that CBR systems are not independent from the regular business (e.g., revision of solutions by the user) and other business applications (e.g., use of customer information in credit assessment). There is currently a lack in considering environmental aspects influencing the maintenance of the knowledge items (e.g., purpose of the CBR system, integration of maintenance into the business process, coordination between maintenance and knowledge management).

The paper extends case-based reasoner maintenance (CBRM) [1] by a more quality-oriented view of the CBR system. First, the quality and quality management concept are introduced. Then, a common framework for structuring relevant environmental aspects and needed elements for a CBR system's quality management (CBRQM) is described. In order to show that CBRM and CBRQM are complementary concepts, selected CBRM approaches and methods are categorized using the elements of the CBRQM framework.

## 2 Defining Case-Based Reasoner Quality Management

As a basis for the framework, the term quality of the knowledge base and case-based reasoner quality management (CBRQM) are characterized.

### 2.1 Quality of the CBR System's Knowledge Base

The knowledge base of a CBR system represents explicit knowledge of a particular application domain. So, the knowledge base can be seen as a representation of the real-world knowledge as perceived by users if the state of the knowledge base at a given time enables the inference of a state of the real-world knowledge (at the same or another time) [2]. This implies that the knowledge base should map the current state of the relevant real-world knowledge.

The performance of the CBR system degrades if inconformities appear between the view of the real-world knowledge that can be inferred from the representing knowledge base and the view that can be obtained by directly observing the real-world knowledge (figure 1) [3].

Inconformities are signs of *quality defects* within the knowledge base. They appear as data defects or knowledge defects. *Data Defects* are all anomalies within the knowledge base that arise from erroneous representations of correct knowledge items, i.e., the knowledge base represents the current state of the real-world knowledge, but there are missing or wrong values. *Knowledge defects* are all anomalies within the knowledge base that can be traced back to unsynchronized knowledge states, i.e., the knowledge items do not have data defects, but they are outdated.

Against this background, the derivation of a quality concept for the knowledge base of CBR systems is guided by the requirements of the users and the

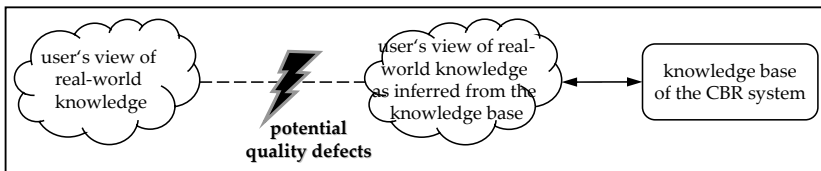


Fig. 1. Quality Defects in CBR Systems

traceability of the quality defects to their sources of appearance and even to the causes of their origins. This corresponds to an information-oriented (user-based) view at the quality of the knowledge base. Based upon the requirements of the users and the state of the real-world knowledge as perceived by the domain experts and the IT-experts, technical requirements for the knowledge containers and knowledge items (product-based view) and the problem solving process (manufacturing-based view) are derived.

The quality concept for the knowledge base of CBR systems is mainly user-based, but also incorporates the other views. Quality can be defined as: a multidimensional measure for the applicability of the knowledge base to fulfil its purpose, whereas this applicability may change over time if data and knowledge defects appear or the underlying needs of the users change [4].

## 2.2 Case-Based Reasoner Quality Management

In its classical meaning quality management is an executive function. Consequently, it is an organizational issue. An efficient quality management should be fixed in all management ranks (normative level, strategic level, and operational level) and established in all structures, activities, and behavioural patterns of the management and the staff [5].

Today's quality management is based upon the ideas of Total Quality Management (TQM) [6]. TQM is an integrated approach for establishing a quality culture within an organization. It is characterized by an enhanced customer orientation, cross-functional cooperation within the organization, the initialization of quality improvements, stronger integration of the people as relevant factor, a proactive quality assurance, and the integration of quality goals into the corporate policy [7]. A TQM-conformant quality management for CBR systems is based upon the three basic principles: customer-, process-, and staff-orientation. It refers to all management activities that constitute the CBR system-relevant aspects of quality policy, goals, and responsibilities, and their transformation into operational planning, controlling, assurance, and improvement processes.

So, case-based reasoner quality management (CBRQM) aims at identifying the expectations and needs of the users, translating them - under consideration of environmental conditions - into requirements for the knowledge items and the problem solving process, and implementing and maintaining the CBR system with respect to maximum user satisfaction [4]. To do so, CBRQM has to encompass all relevant organizational structures, activities and behaviour of the management and the staff.

For operational quality-related structures and activities the process-oriented plan-do-check-act-cycle (PDCA-cycle) has been established [8]. It represents the idea of continuous improvement through the cyclic sequence of quality planning (plan), control (do), assurance (check) and improvement (act). In quality planning the needs of the users are acquired and gradually transferred into guidelines for the design and use of a CBR system (goals, measures, processes, etc.) [9]. During quality control the quality level is checked and verified to hold fixed specifications and to guarantee the mastery of the required processes [9].

Activities in quality assurance and improvement are the selection and execution of maintenance operations. They also encompass CBRM.

The behavioural development supports structures and activities by defining incentives that are able to affect the perceptual and behavioural patterns of the staff interacting with the CBR system and the management.

### 3 A General Framework for CBRQM

The goal of describing the CBRQM framework (figure 2) in this paper is three-fold. First, the framework overviews all basic elements of a methodic quality measurement and control. Second, with the framework a procedure for the design and implementation of quality measurement and control is displayed (denoted by the gray arrows in figure 2). Third, the framework is used as a categorization scheme to exemplify the interrelation between CBRQM and approaches and methods of CBRM.

Using the basic principles of TQM, the *bottom layer* describes general policies of quality management as they are relevant for CBRQM. The *middle layer* contains basic premises and conditions of the system’s environment. They have to be clarified in the run up to the system development or the design of quality measurement and control. They can be understood as a requirement for an efficient operational quality management. The layer distinguishes between managerial guidelines, structures, business activities/processes, and implementation premises and hurdles. The *upper layer* describes the operational elements that are needed to implement a continuous quality measurement and control.

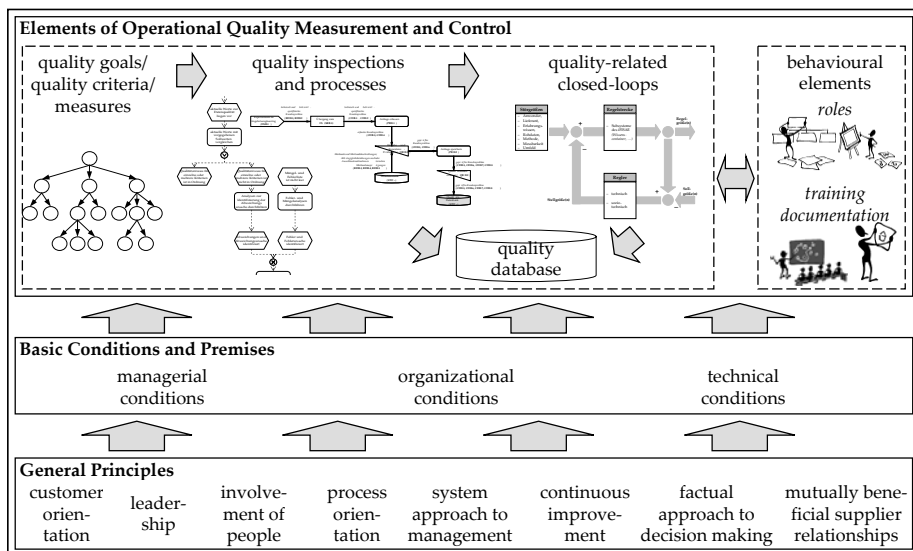


Fig. 2. CBRQM Framework

### 3.1 General Policies

As mentioned above, CBR systems are not independent from the business process in the relevant application domain and the company's application environment. They are applied for a given purpose, e.g. to support help desk operators, for product recommendation, decision making in offer engineering. Against its specific purpose, the CBR system may have strategic importance if its output contributes to an improving company's success.

The intended purpose also determines basic conditions and premises of quality measurement and control. It has to realize the user requirements in the specific domain (*user orientation*). A managerial component is needed. Its tasks are: establishing quality awareness to the staff and the management by communication and coordination, and taking charge of the CBR system's quality management (*leadership*). All members of staff (e.g., users, knowledge suppliers) having a direct or indirect stake in the CBR system should be involved in the quality measurement and control to enable their ability to make real contributions to the success of the system (*involvement of people*).

Due to the fact that the real-world knowledge of an application domain and the quality of the CBR system's knowledge base are dynamic dimensions continual quality measurement and control is needed (*continual improvement*). To realize this requirement, quantifiable information of the current quality level and its history has to be collected and analyzed periodically (*factual approach to decision making*). The tasks of quality measurement and control - collection, assessment, analysis, and modification - are interconnected sequences of activities. Therefore, they must be designed and implemented as processes (*process orientation*). In addition to that, the CBR system and its application domain are part of the overall system "enterprise" and are correlated with other departments and business processes (*system approach to management*).

### 3.2 Basic Conditions and Premises

The conditions and premises of the system's environment are basic for the design and implementation of quality-related activities and the definition of quality criteria for the knowledge base. They are determined by the management, the application domain and parameters of the technical infrastructure.

*Managerial conditions.* The management is responsible for setting the stage for an effective CBRQM, reviewing the effectiveness of quality activities and measures, and intervening if problems arise [11]. For example, managerial conditions force the quality policy for the CBR system. Depending on the strategic importance of a CBR system in the company the policy could call for reactive or proactive activities. Communication and coordination interfaces to other management sub systems (e.g., information management, knowledge management) are defined.

*Organizational conditions.* Organizational conditions are a direct prerequisite for a process-oriented integration of operational CBRQM into the existing business processes of the company. They describe all operational and organizational

structures influencing the design and implementation of quality-related activities and being influenced by these activities.

*Technical conditions.* Beyond organizational conditions, implemented functionality and technical properties of the CBR system and the system platform have to be documented. The functionality gives information about problem solving tasks executed by the system (e.g., only executing retrieval or also executing reuse and revision) and what methods are used (e.g., as retrieval algorithm, adaptation rules). Technical properties describe i.e. data structures used to represent the knowledge items and whether the CBR system is a stand alone-application or an integrated one. They provide indications of the implementation options for measurement, evaluation and modification tasks and methods.

### 3.3 Procedural Elements

In consideration of the basic conditions and premises the upper layer of the framework specifies procedural elements of quality measurement and control.

*Quality goals* are derived from the expectations and needs of the users and the requirements of the CBR system’s quality policy. Out of them, information-oriented and maintenance-oriented *quality criteria* are derived (figure 3).

In the next step, appropriate *quality metrics* and *quality data* are attributed to these quality criteria. They realize the general policies customer orientation and factual approach to decision making. Figure 4 visualizes the derivation of metrics and data using the Goal Question Metric Approach [12], an approach for defining and evaluating a set of operational goals by measurements. The goals are defined in a tractable way by refining them into a set of quantifiable questions

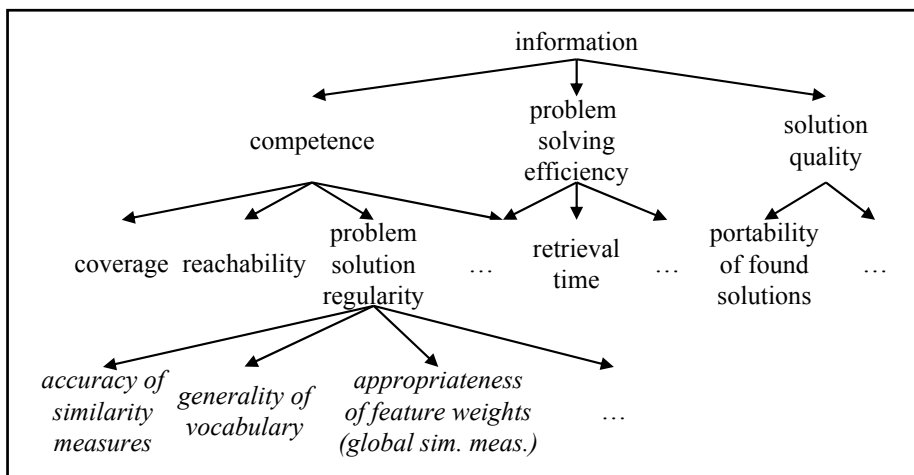


Fig. 3. Pyramid of Needs for Deriving Quality Criteria

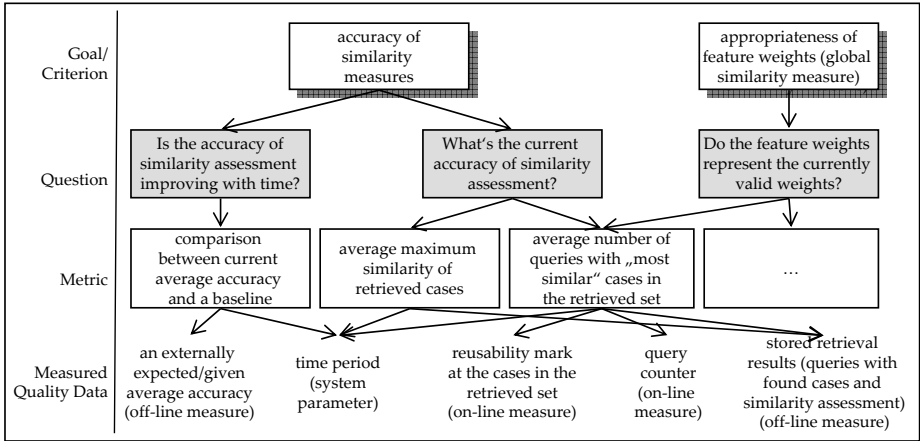


Fig. 4. Deriving Quality Metrics and Data using the Goal Question Metric Approach

that are used to define a specific set of quality metrics and quality data, and provide a framework for interpretation.

*Quality inspections* are used to evaluate the quality level on the basis of the quality criteria. Three types of inspections are differentiated. Static inspections identify the quality of knowledge items at a given point in time, including redundancy checks, inconsistency checks of the case base. Dynamic inspections are used if static inspections do not provide adequate basis for quality evaluations. Here, quality data are collected diachronically over a period of time to display time series of the quality level (e.g., for the accuracy of similarity measures, appropriateness of adaptation rules). A third type of inspection is prompted on deficiencies and alteration lists that enable users and knowledge suppliers to give feedback of detected errors and changed real-world knowledge.

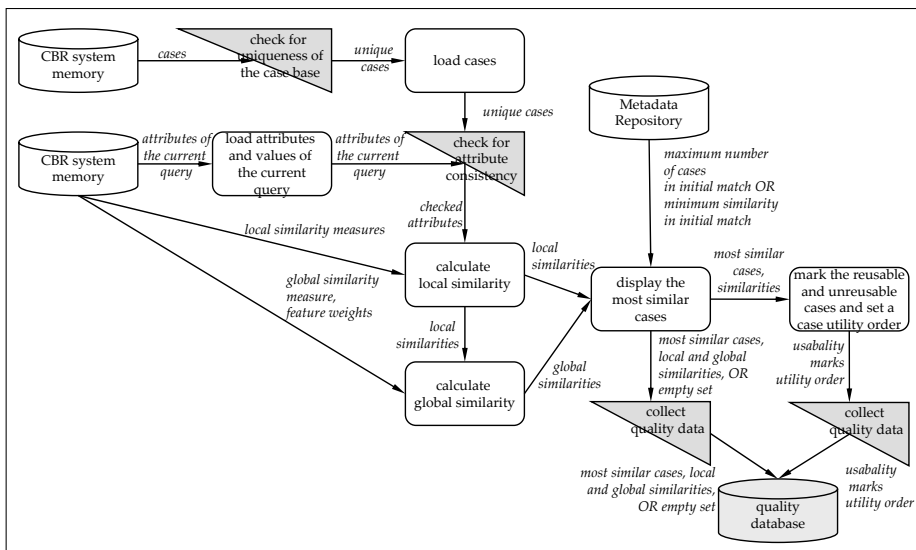
Together with the subsequent activities of initializing and executing maintenance operations the quality inspections are integrated into *quality processes*. Single tasks like the collection of quality data or input checks can be executed during an active problem solving cycle (figure 5). Specific processes are defined like automated test processes, semi-automated and manual collection processes.

Other tasks should be executed in a separate maintenance cycle. In contrast to the application cycle, the activities here are universal (in the majority of cases). Therefore, a common process can be defined including all activities from the quality data collection to the execution of maintenance operations (figure 6).

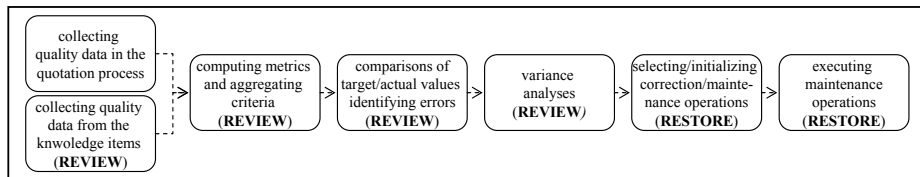
To meet the policy of continual improvement inspections and processes are combined to *quality closed-loops*. For every quality criterion a closed-loop comprises the whole cycle of measurement, evaluation and modification (figure 7).

The administration of quality-related data and information in a *quality database* follows the general policies of process orientation and continual improvement. In particular, the collected data from the dynamic quality inspections and





**Fig. 5.** Example application process *similarity assessment* with a runtime check of case base uniqueness and quality data collection (modified from [4])



**Fig. 6.** Common Quality Process in the Maintenance Cycle

the deficiencies and alterations lists should be recorded in order to decide about maintenance operations and maintenance dates for several quality defects.

### 3.4 Behavioural Elements

The second category on the upper layer is the behavioural elements. They are essential, because social and cultural aspects are key factors of the CBR system’s quality. They are divided in two subcategories. First, incentives and measures for increasing motivation and quality awareness of all people directly and indirectly interacting with the CBR system are needed. They follow the general policies of involvement of people and mutual beneficial supplier relationships. For example, feedback processes should be implemented to provide users and knowledge suppliers an opportunity to point out quality defects. Knowledge suppliers have a stake in the knowledge acquisition processes as they inform about changes in

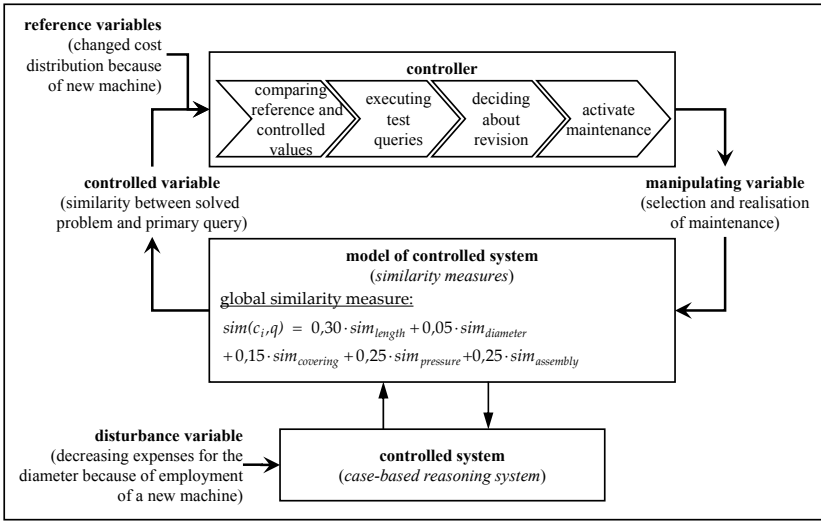


Fig. 7. Closed-Loop Example

relevant real-world knowledge. Users assess the quality of the provided solution suggestions and give feedback to the maintenance staff if they detect data or knowledge defects.

Second, a role and authorization scheme is needed to avoid unclear and overlapping assignment of the maintenance tasks. With the role and authorization concept the policies involvement of people and leadership are realized.

### 4 Integrating CBRM and CBRQM

In order to show that CBRM and the introduced CBRQM framework are not independent concepts, approaches and methods of CBRM are categorized using the elements of the CBRQM framework. Table 1 contains the categories and lists exemplary approaches and methods of CBRM.

*General Policies.* The exemplary contributions do not explicitly concentrate on TQM and its basic principles. But, approaches and basics are selected that could corroborate the ideas of TQM. Here, one can find the experience factory, an approach for organizational learning and reuse of experience in software engineering [14][15], and the quality improvement paradigm, an evolutionary improvement paradigm for software engineering companies [13][12]. Both are basic for INRECA, a methodic approach for the development of industrial CBR systems [16]. Only ROTH-BERGHOFER highlights the relationship between CBRM and Total Quality Management. He requires "maintenance efforts regarding the knowledge of a Case-Based Reasoning system, of course, must be embedded into the quality management system of an organization" ([10], p.37).

**Table 1.** Sample CBRM Approaches and Methods Placed Along the CBRQM Elements

<b>General Principles</b>
<i>TQM principles:</i> e.g., Basili et al. [13,12,14], Althoff et al. [15], Roth-Berghofer [10], Bergmann et al. [16]
<b>Basic premises and conditions</b>
<i>managerial conditions, organizational conditions, technical conditions:</i> e.g., Göker/Roth-Berghofer [17], Bergmann et al. [16]
<b>Procedural and Behavioral Elements</b>
<i>quality goals:</i> e.g., Racine/Yang [18], Smyth/McKenna [20,21,22]
<i>quality criteria, quality metrics, quality data:</i> e.g., Racine/Yang [18], Roth-Berghofer [10], Pal/Shiu [19], Iglezakis [23], Smyth/McKenna [20]
<i>quality inspections, quality processes, quality closed loops:</i> e.g., Leake/Wilson [24,11], Stahl [25], Roth-Berghofer et al. [10,26], Watson [27]
<i>maintenance and knowledge acquisition methods:</i> e.g., Iglezakis [23], Racine/Yang [18,28], Stahl [25,29], Gabel [30], Li et al. [31], Montaner et al. [32], Yang/Zhu [33], Shiu et al. [34], Srinivasan et al. [35]
<i>quality database:</i> e.g. Nick et al. [36,37,38], Menzies [39]
<i>training, management integration:</i> e.g., Roth-Berghofer [10], Nick et al. [36]

*Basic premises and conditions.* The contributions use software process modeling, a complementary approach to the experience factory and the quality improvement paradigm. Similar to the required basic conditions in the CBRQM framework, process models describe managerial processes (e.g., planning and controlling of software engineering projects), organizational processes (e.g. identifying need for changes in the business processes the system should be embedded), and technical processes (e.g., phases of software engineering - analysis, conceptual design) [17,16]. Despite the differences between the specification of these processes and the conditions of the framework, the processes can be starting points for conditions and premises analyzes.

*Procedural Elements - Quality Goals.* To guide maintenance activities so called top-level goals have been introduced [41]. They act as indicators for case base and CBR system performance assessment. Often mentioned goals are:

- effectiveness of the case base as its ability "to answer as many queries as possible efficiently and correctly" [18],
- competence as the range of target problems a given system can solve [20],
- efficiency as the computational cost of solving a set of target problems [21],
- problem solving efficiency as the average problem solving time [22], and
- solution quality as the average quality of a proposed solution [22].

Using these goals as a basis for CBRQM causes the problem that they do not shed light on sources and causes of quality defects. They have to be split up to more fine-grained criteria in order to localize the quality defects.

*Procedural Elements - Quality Criteria.* There are many quality criteria available in the CBRM literature, but mostly for the case base. Most frequently mentioned

are redundancy/uniqueness, consistency, coverage, reachability, correctness, and completeness [18,21,23]. ROTH-BERGHOFER proposes more customized criteria for help desk-applications, e.g, first-call resolution rate, number of escalations needed, and speed up in help-desk operator training [10]. To assess the quality level by the criteria quality data are collected and quality metrics are calculated. In order to collect correct data und calculate right metrics, NICK ET AL. [36,37,38] apply the Goal Question Metric Approach [12,14].

*Procedural Elements - Quality Inspections, Quality Processes, Quality Closed-Loops.* To support the definition of these elements, various approaches/procedures exist. As a start, there is the framework for categorizing maintenance operations by WILSON/LEAKE [24,1]. It describes steps of a common maintenance process from data collection to execution of maintenance operations and policies when and how the steps are executed. As a by-product of his method for learning similarity measures, STAHL extends the steps of the CBR cycle by additional tasks and methods in order to collect data on-line for maintaining the similarity measures. This corresponds with the information by AAMODT/PLAZA that "retain" is the last step in case-based problem solving and learning from experience. ROTH-BERGHOFER ET AL. [10,26] and WATSON recommend the introduction of additional phases – review and reflect [27] or review restore [10] to enable off-line data collection, evaluation and so on. The three approaches may be combined to realize several activities of operational quality evaluation and modification during the application and maintenance cycle.

*Maintenance and Knowledge Acquisition Methods.* They are not a separate element of the CBRQM framework. But, maintenance and knowledge acquisition methods are needed for executing quality assurance and improvement. They are part of the quality evaluation and modification processes. There are pure maintenance methods like data cleansing algorithms to remove redundancies or inconsistencies etc. [18,23,28], algorithms for initial and continual knowledge acquisition [25,29,31,32,33], and algorithms for knowledge base optimization by knowledge transfers between the knowledge containers [30,34,35].

*Procedural Elements - Quality Database.* The term quality database is not often used in CBRM. Past successful and unsuccessful maintenance operations are qualified as maintenance knowledge, and may be stored for reuse in additional knowledge containers. Based upon that assumption, the EMSIG approach describes three knowledge containers for maintenance knowledge [37]: Quality knowledge describes quality goals, criteria, metrics and data to be collected, and records the collected data and the assessed quality criteria. Maintenance process/procedure knowledge defines how maintenance is performed. Maintenance decision knowledge describes under what circumstances maintenance activities should be executed. To realize separate quality databases or additional knowledge containers MENZIES describes meta knowledge, "knowledge about the structure, assessment, or modification of different knowledge types" ([39], p.11). For quality knowledge, he distinguishes between non-functional requirements

(e.g., security, portability), product-oriented assessment knowledge (e.g., for exploration of internal syntactic structures of programs), inconsistency knowledge (for detection of inconsistencies), and critical success metrics (for the contribution of the CBR system to the business success) [39].

*Behavioural Elements - Roles and Responsibilities.* The contributions in the table describe role concepts for two different application domains. NICK ET AL. specify roles and responsibilities for the maintenance of experience bases in software engineering. They name an experience factory manager (defines goals and improvement programs), experience managers (responsible for structure, content and quality of the experience base), project supporter (e.g., recording new experience), experience engineers (e.g., packaging and analyzing existing knowledge), and librarians (technical tasks of maintenance) [36]. ROTH-BERGHOFER describes a role concept for case-based help desk systems. It contains the help desk user (e.g. use the system, reports errors or changes in knowledge), the maintenance engineer (carrying out maintenance operations), knowledge engineer (responsible for continual knowledge acquisition), and administrator (e.g., deciding on maintenance guidelines, scheduling maintenance) [10].

*Behavioural Elements - Training and Management Integration.* There are only few contributions pointing explicit at training programs for users and management integration. In the process models of the INRECA methodology one can find a reference at the design and implementation or the change of training activities for the software engineering project team [17,16]. ROTH-BERGHOFER mentions the need for training, for example with the quality criterion "speedup in help-desk operator training" [10]. With the statement that help-desk operators "must constantly widen and deepen their knowledge. This is encouraged by management with training courses and seminars." ([10], p.118) he includes the importance of management integration.

## 5 Conclusion

In summary, the paper introduced CBRQM as an important enhancement of CBRM for large-scaled and long-term CBR systems in practice. Major elements that play an important role in a CBR system's quality management are summarized in a common framework. The CBRQM framework also describes a procedure for the design and implementation of quality measurement and control. On the basis of exemplary approaches and methods from CBRM it could be pointed out the interrelationship of the two concepts.

Further work may be done with respect to the application of concrete maintenance operations and learning algorithm, in triggering and integrating them into the CBR processes and in user interaction. Secondly, the development of quality databases and of quality knowledge containers should be analyzed in more detail to realize dynamic quality inspections and deficiencies and alteration analyzes.

Finally, a problem of CBR system development and use is still the cost aspect. With the increasing use of these systems in practice, information about the cost

of their maintenance, quality assurance and improvement would lead to more systematic maintenance and, perhaps, better acceptance by the users.

## References

1. Wilson, D.C., Leake, D.B.: Maintaining case-based reasoners: dimensions and directions. In: Computational Intelligence, May 2001, vol. 17, pp. 196–213 (2001)
2. Wand, Y., Wang, R.Y.: Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM* 39(11), 86–95 (1996)
3. Huang, K., Lee, Y.W., Wang, R.Y.: *Quality Information and Knowledge*. Prentice Hall, New Jersey (1999)
4. Bierer, A.: *Qualitätsmessung und -steuerung in Fallbasierten Systemen am Beispiel eines Fallbasierten Systems im Angebotsengineering*. Dissertation Technische Universität Chemnitz, online resource. Chemnitz (2008), <http://archiv.tu-chemnitz.de/pub/2009/0010/index.html>
5. Seghezzi, H.D.: *Integriertes Qualitätsmanagement: das St. Galler Konzept*. Carl Hanser Verlag, München (1996)
6. Deutsches Institut für Normung (DIN) e.V.: *DIN 55350-11. Begriffe zum Qualitätsmanagement – Teil 11: Ergänzung zu DIN EN ISO 9000:2005 (Entwurf)*. Beuth Verlag, Berlin (2007)
7. Helfert, M., Herrmann, C., Strauch, B.: *Datanqualitätsmanagement. Bericht BE HSG/CC DW2/02*, Universität St. Gallen, Institut für Wirtschaftsinformatik. St. Gallen (2001)
8. Deming, W.E.: *Out of the Crisis*. MIT Press, Cambridge (1986)
9. Helfert, M.: *Proaktives Datenqualitätsmanagement in Data-Warehouse-Systemen – Qualitätsplanung und Qualitätslenkung*. Dissertation Universität St. Gallen. Logos Verlag, Berlin (2002)
10. Roth-Berghofer, T.: *Knowledge Maintenance of Case-Based Reasoning Systems. The SIAM Methodology*. Dissertation der Universität Kaiserslautern. Dissertationen zur Künstlichen Intelligenz Nr. 262. Akademische Verlagsgesellschaft Aka GmbH, Berlin (2003)
11. Pfeifer, T.: *Praxishandbuch Qualitätsmanagement*. Hanser Verlag, München (1996)
12. Basili, V.R.: *Software Modeling and Measurement: The Goal/Question/Metric Paradigm*. Technical Report CS-TR-2956. Department of Computer Science, University of Maryland. College Park (1992)
13. Basili, V.R.: *Software Development: A Paradigm for the Future*. In: Knafl, G.J. (ed.) *Proceedings of the 13<sup>th</sup> Annual International Computer Software and Applications Conference*, pp. 471–485. IEEE Computer Society, Washington (1989)
14. Basili, V.R., Caldiera, G., Rombach, H.D.: *The Goal Question Metric Approach*. In: Marciniak, J.J. (ed.) *Encyclopedia of Software Engineering*, vol. 1, pp. 528–532. John Wiley & Sons, Chichester (1994)
15. Althoff, K.-D., Birk, A., Tautz, C.: *The Experience Factory Approach: Realizing Learning from Experience in Software Development Organizations*. IESE-Report No. 013.97/E. Kaiserslautern (1997)
16. Bergmann, R., et al.: *Developing Industrial Case-Based Reasoning Applications. The INRECA Methodology*. Springer, Heidelberg (2003)
17. Göker, M., Roth-Berghofer, T.: *Development and Utilization of a Case-Based Help-Desk Support System in a Corporate Environment*. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) *ICCBR 1999*. LNCS, vol. 1650, pp. 132–146. Springer, Heidelberg (1999)

18. Racine, K., Yang, Q.: On the Consistency Management of Large Case Bases: the Case for Validation. In: Proceedings of the 13th National Conference on Artificial Intelligence, Workshop on Validation and Verification of Knowledge-Based Systems, Portland, Oregon (1996)
19. Pal, S.K., Shiu, S.C.: Foundations of Soft Case-Based Reasoning. Hoboken (2004)
20. Smyth, B., McKenna, E.: Modelling the Competence of Case-Bases. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS, vol. 1488, pp. 208–220. Springer, Heidelberg (1998)
21. Smyth, B., McKenna, E.: Building Compact Competent Case-Bases. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 329–342. Springer, Heidelberg (1999)
22. Smyth, B., McKenna, E.: Footprint-Based Retrieval. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 343–357. Springer, Heidelberg (1999)
23. Iglezakis, I.: Case-Base Maintenance of Case-Based Reasoning Systems in Classification Domains. Dissertation an der Universität Kaiserslautern. Shaker Verlag, Aachen (2004)
24. Leake, D.B., Wilson, D.C.: Categorizing Case-Base Maintenance: Dimensions and Directions. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS, vol. 1488, pp. 196–207. Springer, Heidelberg (1998)
25. Stahl, A.: Learning of Knowledge-intensive Similarity Measures in Case-Base Reasoning. Dissertation Universität Kaiserslautern. Verlag dissertation.de, Berlin (2004)
26. Iglezakis, I., Reinartz, T., Roth-Berghofer, T.: Maintenance Memories: Beyond Concepts and Techniques for Case Base Maintenance. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 227–241. Springer, Heidelberg (2004)
27. Watson, I.: Workshop Summary. CBR Workshop at the 17th International Joint Conference on Artificial Intelligence,  
<http://www.ai-cbr.org/ijcai99/workshop.html> (called: 2006-07-27)
28. Racine, K., Yang, Q.: Redundancy and Inconsistency Detection in Large and Unstructured Case Bases. Technical Report TR97-11. School of Computing Science, Simon Fraser University, Burnaby/CN (1997)
29. Gabel, T., Stahl, A.: Exploiting Background Knowledge when Learning Similarity Measures. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 169–183. Springer, Heidelberg (2004)
30. Gabel, T.: On the Use of Vocabulary Knowledge for Learning Similarity Measures. In: Althoff, K.-D., et al. (eds.) Contributions to the 3<sup>rd</sup> Conference Professional Knowledge Management – Experiences and Visions, April 2005, Kaiserslautern (WM 2005), pp. 253–258. Fraunhofer Publica, Kaiserslautern (2005)
31. Li, Y., et al.: Case-Base Maintenance using Soft Computing Techniques. In: Wang, X., Yeung, D. (eds.) Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, Hebei, China, vol. 4, 5, pp. 1768–1773 (2003)
32. Montaner, M., López, B., de la Rosa, J.L.: Improving case representation and case base maintenance in recommender agents. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416, pp. 234–248. Springer, Heidelberg (2002)
33. Yang, Q., Zhu, J.: A Case-Addition Policy for Case-Base Maintenance. In: Computational Intelligence, May 2001, vol. 17, pp. 250–262 (2001)
34. Shiu, S.C.K., et al.: Transferring Case Knowledge to Adaptation Knowledge: An Approach for Case-Base Maintenance. In: Computational Intelligence, vol. 17, May 2001, pp. 295–314 (2001)

35. Srinivasan, P., et al.: Vocabulary Mining for Information Retrieval: Rough Sets and Fuzzy Sets. *Information Processing & Management* 37(1), 15–38 (2001)
36. Nick, M., Althoff, K.-D.: Systematic Evaluation and Maintenance of Experience Bases. In: Funk, P., Roth-Berghofer, T., Wilson, D.C. (eds.) *Proceedings of the Workshop on Flexible Strategies for Maintaining Knowledge Containers at the 14th European Conference on Artificial Intelligence, Berlin*, pp. 14–21 (2000)
37. Nick, M., Althoff, K.-D.: Acquiring and Using Maintenance Knowledge to Support Authoring for Experience Bases. In: Weber, R., Gresse von Wangenheim, C., Naval Research Laboratory (eds.) *Proceedings of the Workshop Program at the 4th International Conference on Case-Based Reasoning (ICCBR 2001), Washington*, pp. 38–41 (2001)
38. Nick, M., Althoff, K.-D., Jedlitschka, A.: Acquiring Knowledge for Linking Maintenance and Evaluation of Experience Based Information Systems. In: Bergmann, R., Schaaf, M. (eds.) *Proceedings German Workshop on Knowledge and Experience Management (FGWM 2003), Karlsruhe (October 2003)*, [http://km.aifb.uni-karlsruhe.de/ws/LLWA/fgwm/Resources/FGWM03\\_04\\_Markus\\_Nick.pdf](http://km.aifb.uni-karlsruhe.de/ws/LLWA/fgwm/Resources/FGWM03_04_Markus_Nick.pdf)
39. Menzies, T.: Knowledge Maintenance: State of the Art. *Knowledge Engineering Review* 14(1), 1–46 (1999)
40. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7(1), 39–59 (1994)
41. Leake, D.B., Wilson, D.C.: Remembering Why to Remember: Performance-Guided Case-Base Maintenance. In: Blanzieri, E., Portinale, L. (eds.) *EWCBR 2000. LNCS (LNAI), vol. 1898*, pp. 161–172. Springer, Heidelberg (2000)



# Belief Merging-Based Case Combination

Julien Cojan and Jean Lieber

Orpailleur, LORIA, CNRS, INRIA, Nancy University,  
BP 239, 54506 Vandœuvre-lès-Nancy  
{Julien.Cojan, Jean.Lieber}@loria.fr

**Abstract.** Integrity constraint belief merging aims at producing from several knowledge bases, that may be mutually inconsistent, a synthetic knowledge base satisfying a given integrity constraint. It is applied here to case combination for case-based reasoning. This approach is shown to extend Eyke Hüllermeier's credible case-based inference and to be reducible under some assumptions to linear programming.

## 1 Introduction

Case-based reasoning (CBR [1]) aims at solving problems thanks to a set of previously solved problems accompanied with their solutions (the source cases). At least, two general approaches of CBR exist. The first one consists in the adaptation of a sole retrieved case. The second one consists in the combination of  $k \geq 1$  retrieved cases. Actually, case combination is a generalization of case adaptation: this latter is a case combination with  $k = 1$ .

In [2], an approach to adaptation based on a belief revision operator, the so-called conservative adaptation, is presented. A belief revision operator  $\dot{+}$  associates to two knowledge bases  $A$  and  $B$  a knowledge base  $A \dot{+} B$  that entails  $B$  and keeps as much as possible from  $A$ . Conservative adaptation of a source case  $Src$  by a target case  $Tgt$  consists in a revision  $ctxt(Src) \dot{+} ctxt(Tgt)$  where  $ctxt(Src)$  and  $ctxt(Tgt)$  are respectively  $Src$  and  $Tgt$  interpreted according to the domain knowledge.

Now, belief revision is generalized by integrity constraint (IC) belief merging [3] that integrates several knowledge bases altogether with constraints.

The purpose here is to substitute belief revision by IC belief merging in conservative adaptation to define a case combination approach.

After the introduction of a running example and a preliminary section, the IC belief merging theory is presented in section 4 and its application to case combination in section 5. Section 6 shows that credible case-based inference [4] is a kind of case combination. The computation of belief merging in numerical spaces is studied in section 7, before a related work review and the conclusion.

## 2 Introduction of the Running Example

Assume you have an egg allergic guest and you have the experience of some dishes that can be made without eggs. Your guest loves chocolate and chocolate mousse would be

a perfect desert, even if you don't have an egg-free recipe at disposal. Several recipes –the source cases  $\text{Srce}_i$ – can be combined to solve the target case  $\text{Tgt}$ :

$\text{Tgt}$ : egg-free chocolate mousse recipe

$\text{Srce}_1$ : chocolate mousse recipe

$\text{Srce}_2$ : egg-free Chantilly recipe

$\text{Srce}_3$ : egg-free chocolate cream recipe

The expected solution would be to follow the main lines from  $\text{Srce}_1$  but to substitute the egg-snow by egg-free Chantilly from  $\text{Srce}_2$  and the chocolate mix –that contains egg yolks– by the chocolate cream from  $\text{Srce}_3$ .

### 3 Preliminaries

#### 3.1 Set Theory Notations

A boolean interpretation  $\mathcal{I}$  on a set of variables  $\mathcal{V} = \{a_1, a_2, \dots, a_n\}$  is a mapping from  $\mathcal{V}$  to the set of boolean values  $\mathbb{B} = \{\text{true}, \text{false}\}$ . It can be assimilated to  $(x_1, \dots, x_n) \in \mathbb{B}^n$  where  $\mathcal{I}(a_i) = x_i$  for  $i = 1$  to  $n$ . Thus, the models  $[f]$  of a formula  $f$  –the set of the interpretations satisfying  $f$ – is assimilated to a subset of  $\mathcal{U} = \mathbb{B}^n$ . Following the *principle of irrelevance of syntax*, the formulas are assimilated in this paper to their models and thus to subsets of  $\mathcal{U}$ . In particular conjunction  $\wedge$ , entailment  $\models$ , and logical equivalence  $\equiv$  represent intersection  $\cap$ , inclusion  $\subseteq$ , and set equality  $=$  on subsets of  $\mathcal{U}$ . More precisely,  $[f \wedge g] = [f] \cap [g]$ ,  $f \models g$  iff  $[f] \subseteq [g]$ , and  $f \equiv g$  iff  $[f] = [g]$ .

Propositional logic can then be generalized considering any set  $\mathcal{U}$ , in particular attribute-values formalisms correspond to sets of the kind  $\mathcal{U} = D_1 \times \dots \times D_n$  where  $D_1, \dots, D_n$  are more elementary sets like integers  $\mathbb{Z}$ , positive real numbers  $\mathbb{R}^+$ , etc.

In order to ease the reading, a variable  $x_i$  that can take any value from  $D_i$  is just written ' $\_$ ', eg. if  $n = 3$  and  $a_1 \in D_1$ ,  $a_3 \in D_3$ ,  $\{(a_1, \_, a_3, \_)\} = \{(a_1, x_2, a_3) \mid x_2 \in D_2\}$ . So,  $\mathcal{U} = \{(\_, \_, \_)\}$ .

#### 3.2 Metric Spaces

**Definition 1.** A pseudo-distance<sup>1</sup> on a set  $\mathcal{U}$  is a function  $d : \mathcal{U} \times \mathcal{U} \rightarrow [0; +\infty]$  satisfying the separation axiom:

$$\text{for any } x, y \in \mathcal{U}, d(x, y) = 0 \text{ iff } x = y$$

A distance on  $\mathcal{U}$  is a pseudo-distance on  $\mathcal{U}$  taking only finite values (for any  $x, y \in \mathcal{U}$ ,  $d(x, y) < +\infty$ ) that satisfies the symmetry axiom:

$$\text{for any } x, y \in \mathcal{U}, d(x, y) = d(y, x)$$


and the triangular inequality:

<sup>1</sup> We slightly abuse words here as this may not be in agreement with common definition of pseudo-distance.

for any  $x, y, z \in \mathcal{U}$ ,  $d(x, z) \leq d(x, y) + d(y, z)$

A (pseudo-)metric space is a pair  $(\mathcal{U}, d)$  where  $d$  is a (pseudo-)distance on the set  $\mathcal{U}$ .

$2^{\mathcal{U}}$  denotes the set of subsets of  $\mathcal{U}$ . A pseudo-distance  $d$  on  $\mathcal{U}$  is extended on subsets of  $\mathcal{U}$  as follows:



$$d(A, y) = \inf_{x \in A} d(x, y)$$

$$d(A, B) = \inf_{x \in A, y \in B} d(x, y) = \inf_{y \in B} d(A, y)$$

where  $A, B \in 2^{\mathcal{U}}$ , and  $x, y \in \mathcal{U}$  (note that  $d : 2^{\mathcal{U}} \times 2^{\mathcal{U}} \rightarrow [0; +\infty]$  is not necessary a pseudo-distance).

$A \in 2^{\mathcal{U}}$  is bounded if there exists  $K \in \mathbb{R}^+$  such that for each  $x, y \in A$ ,  $d(x, y) \leq K$ . Given a pseudo-distance on  $\mathcal{U}$ ,  $x, y \in \mathcal{U}$  and  $r \in [0; +\infty]$ , the *right closed ball of center  $x$  and radius  $r$*  is the set  $\mathcal{B}_r^{\mathcal{U}}(x) = \{y \in \mathcal{U} \mid d(x, y) \leq r\}$  and the *left closed ball of center  $y$  and radius  $r$*  is the set  $\mathcal{B}_r^{\mathcal{U}}(y) = \{x \in \mathcal{U} \mid d(x, y) \leq r\}$ . The distinction between these two definitions is relevant when  $d$  is not symmetrical, otherwise they are equal.

**Definition 2.** A (pseudo-)discrete metric space  $(\mathcal{U}, d)$  is a (pseudo-)metric space such that, for any  $x \in \mathcal{U}$  and  $r \in [0; +\infty[$ ,  $\mathcal{B}_r^{\mathcal{U}}(x)$  and  $\mathcal{B}_r^{\mathcal{U}}(x)$  are finite.

This definition is stronger than the usual definition of (pseudo-)discrete metric space which states that for any  $x \in \mathcal{U}$ , there is an  $r > 0$  such that  $\mathcal{B}_r^{\mathcal{U}}(x) = \mathcal{B}_r^{\mathcal{U}}(x) = \{x\}$ .

If  $(\mathcal{U}, d)$  is discrete then for  $A, B$  two subsets of  $\mathcal{U}$ , if  $A$  is bounded, then  $A$  is finite and  $d(A, B) = d(A, y)$  for some  $y \in B$ . Moreover  $\{z \in \mathcal{U} \mid d(A, z) = 0\} = A$  (i.e.  $A$  is closed).

To avoid continuity issues, like no minimum for a distance, only discrete spaces will be considered in the following. In particular  $\mathbb{R}$  will be approximated by decimals of a fixed maximum length.

### 3.3 CBR: Definitions and Hypotheses

*Cases and domain knowledge.* Case-based reasoning (CBR) aims at solving problems of a given application domain with the help of previous solving episodes, or *cases*. In this paper, these notions are formalized as follows. Let  $\mathcal{U}_{\text{pb}}$  and  $\mathcal{U}_{\text{sol}}$  be two sets:  $x \in \mathcal{U}_{\text{pb}}$  is a *problem instance*,  $X \in \mathcal{U}_{\text{sol}}$  is a *solution instance*. A *problem pb* is a class of problem instances:  $\text{pb} \in 2^{\mathcal{U}_{\text{pb}}}$ . A *solution sol* is a class of solution instances:  $\text{sol} \in 2^{\mathcal{U}_{\text{sol}}}$ .

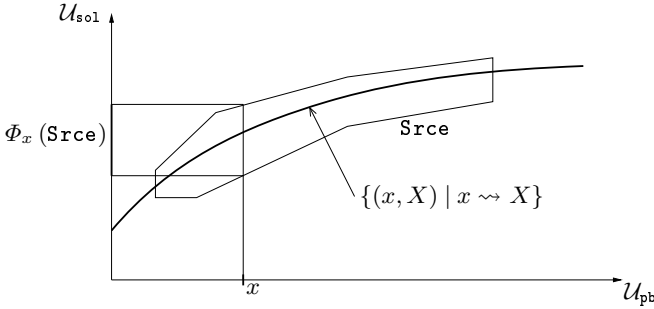
Let  $\mathcal{U} = \mathcal{U}_{\text{pb}} \times \mathcal{U}_{\text{sol}}$ . A *case* is a subset of  $\mathcal{U}$ . A *singleton case* is a case with only one element:  $\text{Case} = \{(x, X)\}$ . Given  $x$  and  $\text{Case}$ , a problem instance and a case,  $\Phi_x(\text{Case})$  denotes the projection of  $x$  on solution instances:

$$\Phi_x(\text{Case}) = \{X \in \mathcal{U}_{\text{sol}} \mid (x, X) \in \text{Case}\}$$

In particular, if  $\text{Case} = \{(x, X)\}$  is a singleton case then  $\Phi_x(\text{Case}) = \{X\}$ .

The relationship stating that a solution  $sol$  solves a problem  $pb$  is formalized by a binary relation  $\rightsquigarrow$  on  $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{sol}$ :  $pb$  is solved by  $sol$  if, for every  $x \in pb$  exists  $X \in sol$  such that  $x \rightsquigarrow X$ .

$\rightsquigarrow$  is not assumed to be completely known. By contrast, it is assumed that a finite set of cases, the *case base*  $CB$  and some domain knowledge are available, and that any case of the case base  $CB$  –called a *source case* and denoted by  $Srce$ – has the following property: for each  $x \in \mathcal{U}_{pb}$ , if  $\Phi_x(Srce) \neq \emptyset$  then there exists  $X \in \Phi_x(Srce)$  such that  $x \rightsquigarrow X$  (see figure **1**).



**Fig. 1.** A source case  $Srce$

The target case, denoted by  $Tgt$ , is the case for which the solution part has to be made more precise by the current CBR session. In general, before the CBR inference, nothing is known about this solution:  $Tgt = tgt \times \mathcal{U}_{sol}$  with  $tgt \in 2^{\mathcal{U}_{pb}}$ , the *target problem* (Fig. **2**). A *singleton target problem* is a target problem with only one element:  $tgt = \{x_0\}$ .

The *domain knowledge* states that some pairs  $(x, X)$  are not licit:  $x \not\rightsquigarrow X$ . Thus, it corresponds to a necessary condition for  $x \rightsquigarrow X$ . It is formalized by a subset  $DK$  of  $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{sol}$  and satisfies the implication:  $x \rightsquigarrow X$  implies  $(x, X) \in DK$  (or, by contraposition,  $(x, X) \notin DK$  implies  $x \not\rightsquigarrow X$ ). For *Case*, a given case, its elements  $(x, X)$  that are not consistent with  $DK$  have not to be considered (they are known to be illicit). Thus, *Case* is to be considered in conjunction with  $DK$ , i.e. in its context  $ctxt(Case) = DK \cap Case$ . If no domain knowledge is available, then  $DK = \mathcal{U}$ : every pair  $(x, X) \in \mathcal{U}$  is *a priori* licit.

*Case-based inference.* Given a target case  $Tgt$ , a case base  $CB$  and the domain knowledge  $DK \subseteq \mathcal{U}$ , the case-based inference aims at proposing a case  $SolvedTgt$  that makes  $Tgt$  more precise (Fig. **2**):

$$DK \cap SolvedTgt \subseteq DK \cap Tgt \tag{1}$$

Two main approaches for this inference are described in the CBR literature. The first one is based on adaptation and the second one on combination.

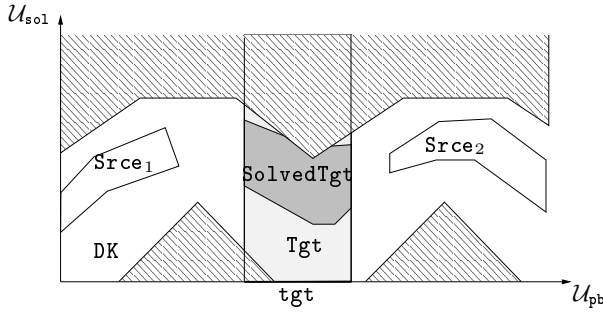


Fig. 2. Result of a CBR session, Tgt has been specialized into SolvedTgt

### 3.4 Formalization of the Example

The cooking problem specification consists in the three following conditions: whether the dish is frothy, whether it contains eggs, and whether it contains chocolate.  $\mathcal{U}_{pb} = \mathcal{U}_1 \times \mathcal{U}_2 \times \mathcal{U}_3$ .

Only the ingredient amounts and the volume of froth will be considered for the solution. All the values are taken for a single serving –which explains the real values for the eggs number.  $\mathcal{U}_{so1} = \mathcal{U}_4 \times \mathcal{U}_5 \times \dots \times \mathcal{U}_9$ . And finally  $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{so1}$ . The cases values are given in table 1.

Table 1. Formalization of the egg free chocolate mousse example

	attribute	Tgt	Srce <sub>1</sub>	Srce <sub>2</sub>	Srce <sub>3</sub>
$\mathcal{U}_1 = \mathbb{B}$	is frothy	true	true	true	false
$\mathcal{U}_2 = \mathbb{B}$	has eggs	false	true	false	false
$\mathcal{U}_3 = \mathbb{B}$	has chocolate	true	true	false	true
$\mathcal{U}_4 = \mathbb{R}$	froth volume (ml)	–	200	225	0
$\mathcal{U}_5 = \mathbb{R}$	number of eggs	–	0.67	0	0
$\mathcal{U}_6 = \mathbb{R}$	chocolate mass (g)	–	35	0	25
$\mathcal{U}_7 = \mathbb{R}$	cream mass (g)	–	10	0	125
$\mathcal{U}_8 = \mathbb{R}$	sugar mass (g)	–	65	50	–
$\mathcal{U}_9 = \mathbb{R}$	soya volume (ml)	–	0	170	0

The domain knowledge is given by DK:

$$DK = R_{vFroth} \cap R_{hasFroth} \cap R_{hasEggs} \cap R_{hasChocolate}$$

where  $R_{vFroth}$  states that the froth obtained from an egg  $v$  is at most 200 ml and 220 ml from 170 ml of soya milk ( $220/170 \simeq 1.32$ ):

$$R_{vFroth} = \{(\_, \_, \_, vFroth, eggs, \_, \_, \_, soya) \mid vFroth \leq 200 \times eggs + 1.32 \times soya\}$$

$R_{\text{hasFroth}}$ ,  $R_{\text{hasEggs}}$ , and  $R_{\text{hasChocolate}}$  force `hasFroth` (resp. `hasEggs` and `hasChocolate`) to be true only when there is froth (resp. eggs and chocolate) in the recipe:

$$\begin{aligned}
 R_{\text{hasFroth}} &= \{(\text{hasFroth}, \_, \_, \text{vFroth}, \_, \_, \_, \_, \_) \mid \\
 &\qquad \qquad \qquad \text{hasFroth} = \text{true iff vFroth} \neq 0\} \\
 R_{\text{hasEggs}} &= \{(\_, \text{hasEggs}, \_, \_, \text{eggs}, \_, \_, \_, \_) \mid \text{hasEggs} = \text{false iff eggs} = 0\} \\
 R_{\text{hasChocolate}} &= \{(\_, \_, \text{hasChocolate}, \_, \_, \text{chocolate}, \_, \_, \_) \mid \\
 &\qquad \qquad \qquad \text{hasChocolate} = \text{false iff chocolate} = 0\}
 \end{aligned}$$

Therefore, the fifth component of  $\text{DK} \cap \text{Tgt}$  is  $(\text{DK} \cap \text{Tgt})_5 = 0$ .

The following distance  $d$  is defined on  $\mathcal{U}$ , for  $x, y \in \mathcal{U}$ , by:

$$\begin{aligned}
 d(x, y) &= d_{\text{pb}}(x, y) + d_{\text{so1}}(x, y) \\
 d_{\text{pb}}(x, y) &= \sum_{i=1}^3 w_i \times \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases} \\
 d_{\text{so1}}(x, y) &= \sum_{i=4}^9 w_i |y_i - x_i|
 \end{aligned}$$

The choice of the weights  $w_i$  reflects the relative importance of the different dimensions. In particular here, the eggs and soya milk are used to generate froth, thus froth's dimension should get a higher importance than egg's and soya's.

## 4 Integrity Constraint Belief Merging

The following scenario illustrates the notion of integrity constraint belief merging. Let us consider an agent that has some knowledge about the world that he/she considers to be inviolable: this is his/her integrity constraints (IC). Now, he/she receives from several sources some knowledge about the world. Taking the conjunction of all these sources does not necessarily lead to a knowledge base consistent with the IC of the agent.

Various operators may be used to merge these sources of knowledge in a result consistent with the integrity constraints. Such an IC merging operator should satisfy some postulates [3], as it is explained in section 4.1. A straightforward generalization of this work to a more general formalism is presented in section 4.2. Then, an example of IC merging operator is presented (section 4.3).

### 4.1 IC Merging in Propositional Logic

The definition below is a reformulation from [3], with substitution of propositional formulas on  $\mathcal{V}$  by subsets of  $\mathbb{B}^n$ :

**Definition 3.** *Let  $\mathcal{U} = \mathbb{B}^n$ . An IC merging operator on  $\mathcal{U}$  is a mapping  $\Delta : (IC, M) \mapsto \Delta_{IC}(M)$ , where  $IC \in 2^{\mathcal{U}}$  is the integrity constraint and  $M$  –the set of beliefs to be merged– is a finite multi-set of non empty subsets of  $\mathcal{U}$ , satisfying the following postulates:*

- ( $\Delta$ -1)  $\Delta_{IC}(M) \subseteq IC$   
 ( $\Delta$ -2) If  $IC \neq \emptyset$  then  $\Delta_{IC}(M) \neq \emptyset$   
 ( $\Delta$ -3) If  $\bigcap M \cap IC \neq \emptyset$  then  $\Delta_{IC}(M) = \bigcap M \cap IC$   
 ( $\Delta$ -4) If  $A_1 \subseteq IC$  and  $A_2 \subseteq IC$  then  
 $\Delta_{IC}(\{A_1, A_2\}) \cap A_1 \neq \emptyset$  iff  $\Delta_{IC}(\{A_1, A_2\}) \cap A_2 \neq \emptyset$   
 ( $\Delta$ -5)  $\Delta_{IC}(M_1) \cap \Delta_{IC}(M_2) \subseteq \Delta_{IC}(M_1 \cup M_2)$   
 ( $\Delta$ -6) If  $\Delta_{IC}(M_1) \cap \Delta_{IC}(M_2) \neq \emptyset$  then  $\Delta_{IC}(M_1 \cup M_2) \subseteq \Delta_{IC}(M_1) \cap \Delta_{IC}(M_2)$   
 ( $\Delta$ -7)  $\Delta_{IC_1}(M) \cap IC_2 \subseteq \Delta_{IC_1 \cap IC_2}(M)$   
 ( $\Delta$ -8) If  $\Delta_{IC_1}(M) \cap IC_2 \neq \emptyset$  then  $\Delta_{IC_1 \cap IC_2}(M) \subseteq \Delta_{IC_1}(M)$

where  $IC, IC_1, IC_2, A_1, A_2 \in 2^{\mathcal{U}}$ ,  $A_1 \neq \emptyset$ ,  $A_2 \neq \emptyset$  and  $M, M_1$ , and  $M_2$  are three multisets of non empty subsets of  $\mathcal{U}$ .  $M_1 \cup M_2$  denotes the multi-set union of  $M_1$  and  $M_2$ .

Note that there is another postulate in [3] that expresses the principle of irrelevance of syntax. By working on interpretations the independence to the syntax is already implied, thus this postulate is reformulated in the following tautology: if  $M_1 = M_2$  and  $IC_1 = IC_2$  then  $\Delta_{IC_1}(M_1) = \Delta_{IC_2}(M_2)$ .

*IC merging and belief revision.* Belief revision is usually presented as the change of an agent belief  $A$  after some facts  $B$  are known by him. Some beliefs in  $A$  may have to be left as being in contradiction with  $B$  but others may not have interference and should be kept. The resulting belief  $A \dot{+} B$  should entail  $B$  and keep “as much as possible” of  $A$ . The notion of IC merging can be considered as a generalization of the notion of revision in the sense that if  $\dot{+}$  is defined by

$$A \dot{+} B = \Delta_B(\{A\}) \quad (2)$$

with  $\Delta$  an IC merging operator, then  $\dot{+}$  satisfies the postulates of revision (i.e., the postulates of the “AGM theory” [5], that have been applied to propositional logic in [6]).

## 4.2 Generalization

The definition of an IC merging operator on a given set  $\mathcal{U}$  is the same as definition 3 except that  $\mathcal{U}$  is any set, not necessarily  $\mathbb{B}^n$ .

**Definition 4.** A pre-IC merging operator on a set  $\mathcal{U}$  is a mapping  $\Delta : (IC, M) \mapsto \Delta_{IC}(M)$ , where  $IC \in 2^{\mathcal{U}}$  and  $M$  is a finite multi-set of subsets of  $\mathcal{U}$ , satisfying the postulates ( $\Delta$ -1), ( $\Delta$ -3) to ( $\Delta$ -8), and ( $\Delta$ -2') a weakened version of ( $\Delta$ -2):

$$(\Delta\text{-2}') \text{ If } IC \neq \emptyset \text{ and every set in } M \text{ is bounded, then } \Delta_{IC}(M) \neq \emptyset$$

**Definition 5.** An IC merging operator is a pre-IC merging operator that satisfies postulate ( $\Delta$ -2).

### 4.3 Example of Pre-IC Merging Operator

The operator presented in this section is inspired from one of the  $DA^2$  operators [3].

Let  $(\mathcal{U}, d)$  be a pseudo-metric space. Let  $d^\Sigma$  be the function associating to the pair  $(M, y)$  –where  $M$  is a finite multiset of subsets of  $\mathcal{U}$ , and  $y \in \mathcal{U}$ – the real number

$$d^\Sigma(M, y) = \sum_{A \in M} d(A, y)$$

For  $IC \in 2^\mathcal{U}$ , let  $d^\Sigma(M, IC) = \inf_{y \in IC} d^\Sigma(M, y)$ . Let  $\Delta^{d, \Sigma}$  be the operator defined as follows:

$$\Delta_{IC}^{d, \Sigma}(M) = \{y \in IC \mid d^\Sigma(M, y) = d^\Sigma(M, IC)\}$$

**Proposition 1.** *If  $(\mathcal{U}, d)$  is discrete,  $\Delta^{d, \Sigma}$  satisfies the postulates  $(\Delta-1)$ ,  $(\Delta-2')$ ,  $(\Delta-3)$ ,  $(\Delta-5)$ ,  $(\Delta-6)$ ,  $(\Delta-7)$ , and  $(\Delta-8)$ .*

*If  $d$  is symmetrical (i.e., it satisfies the symmetry axiom) then  $\Delta^{d, \Sigma}$  satisfies  $(\Delta-4)$ . It is then a pre-IC merging operator.*

A proof for this proposition is given in appendix.

Note, that  $(\Delta-2)$  is not satisfied in general. Consider  $\mathcal{U} = \{\log(n) \mid n \in \mathbb{N} \setminus \{0\}\}$  with  $d(x, y) = |y - x|$ ,  $IC = \{\log(2p) \mid p \in \mathbb{N} \setminus \{0\}\}$ ,  $M = \{A\}$  with  $A = \{\log(2p+1) \mid p \in \mathbb{N}\}$ . Then  $\Delta_{IC}^{d, \Sigma}(M) = \emptyset$ .

## 5 Case Combination Based on a Pre-IC Merging Operator

### 5.1 Conservative Adaptation

Conservative adaptation [2] is an approach to adaptation based on belief revision. Its principle is to reuse “as much as possible” of  $Srce$  while being consistent with  $Tgt$ . Both  $Srce$  and  $Tgt$  must be considered according to domain knowledge  $DK$ . As for belief revision, the meaning of “as much as possible” is variable. The idea is to define conservative adaptation parameterized by a belief revision operator  $\dot{+}$ :

$$SolvedTgt = (DK \cap Srce) \dot{+} (DK \cap Tgt)$$

This inference is called  $\dot{+}$ -conservative adaptation.

### 5.2 $\Delta$ -Combination of Cases

*Definition.* Let  $SCS = \{Srce_1, \dots, Srce_k\}$  be a subset of the case base  $CB$ . Let  $\Delta$  be a pre-IC merging operator on  $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{so1}$ .  $\Delta$ -combination of cases is a generalization of  $\dot{+}$ -conservative adaptation:  $\dot{+}$  is generalized in  $\Delta$  and the sole selected case is generalized in the set of source cases  $SCS$ . Thus,  $SolvedTgt = CC_{\Delta}^{DK}(SCS, Tgt)$  with

$$CC_{\Delta}^{DK}(\{Srce_1, \dots, Srce_k\}, Tgt) = \Delta_{DK \cap Tgt}(\{DK \cap Srce_1, \dots, DK \cap Srce_k\}) \quad (3)$$

I.e., the contribution of the source cases are merged in a result that specializes the target case.

If  $k = \text{card}(SCS) = 1$ , then  $CC_{\Delta}^{DK}(SCS, Tgt)$  is a  $\dot{+}$ -conservative adaptation, with  $\dot{+}$  defined by [2].



*Properties.* In this section, the consequences of the postulates of (pre-)IC merging operators are discussed from a  $\Delta$ -combination of cases viewpoint.

( $\Delta$ -1) entails that  $DK \cap \text{SolvedTgt} \subseteq DK \cap \text{Tgt}$  which is the property **(I)** required for the case-based inference (cf. section **3.3**).

( $\Delta$ -2') entails that if the target case is consistent with the domain knowledge  $-DK \cap \text{Tgt} \neq \emptyset$  and each source case is bounded, -e.g., singleton cases- then  $DK \cap \text{SolvedTgt}$  is satisfiable. ( $\Delta$ -2) is stronger as it does not require the source cases to be bounded.

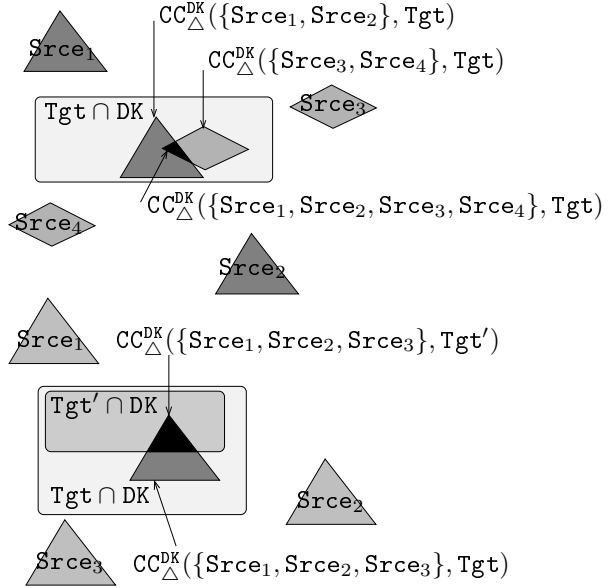
( $\Delta$ -3) entails that if the cases of SCS are consistent altogether with  $DK$ , then  $\text{SolvedTgt}$  is the conjunction of  $DK$ ,  $\text{Tgt}$ , and every  $\text{Srce} \in \text{SCS}$ .

( $\Delta$ -4) enforces equity between the source cases: if two source cases are consistent with the target context, then either both of them are taken into account in the combination or none of them.

( $\Delta$ -5) to ( $\Delta$ -8) characterize the maximal preservation of the source cases according to the local decomposition of SCS and  $\text{Tgt}$ .

( $\Delta$ -5) and ( $\Delta$ -6) state that if the combination of two subsets of SCS provide consistent solutions, then the combination of the whole is the conjunction of both solutions.

( $\Delta$ -7) and ( $\Delta$ -8) state that if  $\text{Tgt}$  is specialized into  $\text{Tgt}'$  that is consistent with the case combination for  $\text{Tgt}$ , then the case combination for  $\text{Tgt}'$  is obtained by conjunction with  $\text{Tgt}'$ .



### 5.3 Application to the Example

Consider the merging operator defined in section **4.3** using the distance  $d$  defined in section **3.4**. The values of dimensions  $U_i$  for  $i = 1, 2, 3, 5$  are fixed in  $DK \cap \text{Tgt}$ , so the space to be explored for the minima of  $d^{\Sigma}$  corresponds to  $i = 4, 6, 7, 8, 9$ .

From the structure of  $d$ , it can be shown that the minimum can be searched independently for dimensions  $U_i$  with  $i = 6, 7, 8$  as there is no constraint relating their values. Then the minima does not depend on the weight  $w_i$ , it is reached for:

$$x_6 = 25 \quad x_7 = 10 \quad 50 \leq x_8 \leq 65$$

$U_4, U_5$ , and  $U_9$  are related through  $R_{v\text{Froth}}$ , the search must then follow this restriction:  $x_4 \leq 200 \times x_5 + 1.32 \times x_9$ . As  $x_5 = 0$  it becomes:  $x_4 \leq 1.32 \times x_9$ .

As seen in section 3.4, the volume of froth should be given priority over the ways to making it. From the structure of  $d^\Sigma$  used in  $\Delta^{d,\Sigma}$ , it can be shown that the condition  $w_4 > 3 \times (w_9 + w_5)$  is enough to ensure that priority (3 for the number of cases and  $w_9, w_5$  because these are the dimensions in competition). Under this assumption, the minima is obtained for:

$$x_4 = 200 \qquad x_9 = 165$$

This result matches with what was expected in section 2: the froth volume and chocolate mass are close to those of  $\text{Srce}_1$ . The use of soya milk to generate froth is taken from  $\text{Srce}_2$  with an adaptation according to the froth volume.  $\text{Srce}_3$  had little influence, however its selection as source case offsets the absence of chocolate in  $\text{Srce}_2$ .

## 6 Application to CCBI

### 6.1 Credible Case-Based Inference

*Assumptions.* Credible case-based inference (CCBI [4]) is an approach to CBR for which the problem-solution relation is assumed to be a partial function: if  $x \rightsquigarrow X$  and  $x \rightsquigarrow X'$  then  $X = X'$ . Moreover, each source case is a singleton  $\text{Srce}_i = \{(x_i, X_i)\}$  and the target case is specified by a singleton target problem:  $\text{Tgt} = \{x_0\} \times \mathcal{U}_{\text{so1}}$ .

CCBI is based on the idea that the CBR principle “Similar problems have similar solutions” can be modeled thanks to  $d_{\text{pb}}$ , a symmetrical pseudo-distance on  $\mathcal{U}_{\text{pb}}$ ,  $d_{\text{so1}}$ , a symmetrical pseudo-distance on  $\mathcal{U}_{\text{so1}}$ , and  $h$ , a *similarity profile*, i.e., a function  $h : [0; +\infty] \rightarrow [0; +\infty]$  such that for *most*  $x, y \in \mathcal{U}_{\text{pb}}$ , if  $x \rightsquigarrow X$  and  $y \rightsquigarrow Y$  then

$$d_{\text{so1}}(X, Y) \leq h(d_{\text{pb}}(x, y)) \tag{4}$$

Thus, the similarity between solutions is constrained by the similarity between problems [2]. A way to learn  $h$  from the case base is also described in [4] and it is proven, under technical assumptions, that the probability of having the constraint (4) violated converges to 0 as the size of the case base grows.

*Definition of CCBI.* Given a set  $\text{SCS}$  of source cases  $\text{Srce}_i = \{(x_i, X_i)\}$  and a target problem  $x_0$ , if  $x = x_i$  and  $y = x_0$  satisfies (4) for any  $i$ , then the solution  $X_0$  of  $x_0$  satisfies  $d_{\text{so1}}(X_i, X_0) \leq h(d_{\text{pb}}(x_i, x_0))$ . In other words (Fig. 3):

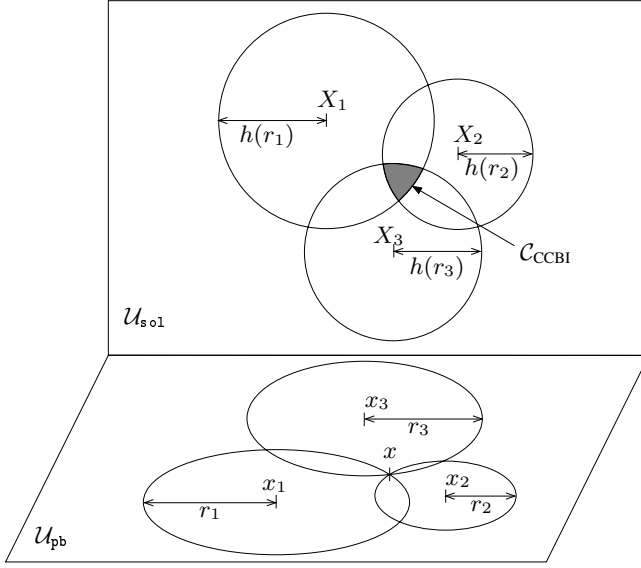
$$X_0 \in \mathcal{C}_{\text{CCBI}} = \bigcap_i \mathcal{B}_{h(d_{\text{pb}}(x_i, x_0))}^{\mathcal{U}_{\text{so1}}}(X_i) \tag{5}$$

Therefore,  $\text{SolvedTgt} = \{x_0\} \times \mathcal{C}_{\text{CCBI}}$  solves  $\text{Tgt}$ .

This inference is only credible and not certain since (4) is only satisfied for *most*  $x, y \in \mathcal{U}_{\text{pb}}$ .

---

<sup>2</sup> In fact, CCBI is introduced in [4] thanks to similarity measures  $\mathcal{S}_{\text{pb}}$  and  $\mathcal{S}_{\text{so1}}$  on  $\mathcal{U}_{\text{pb}}$  and  $\mathcal{U}_{\text{so1}}$ , but the definition presented in the current paper is equivalent. Indeed, a similarity measure  $\mathcal{S}$  on  $\mathcal{U}$  verifying  $\mathcal{S}(x, y) = 1$  iff  $x = y$  can be defined thanks to a pseudo-distance  $d$  on  $\mathcal{U}$  by  $\mathcal{S}(x, y) = \frac{1}{1+d(x, y)}$  and vice-versa.



**Fig. 3.** Credible Case-Based Inference

## 6.2 $\Delta^{d,\Sigma}$ -Combination of Cases Extends CCBI

**Proposition 2.** *CCBI assumption about the case base is made (cf. section 6.1).*

Let  $d$  be the pseudo-distance on  $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{so1}$  defined for  $(x, X), (y, Y) \in \mathcal{U}$  by

$$d((x, X), (y, Y)) = \max\{h(d_{pb}(x, y)), d_{so1}(X, Y)\}$$

Let  $\mathcal{C}_{CCBI}$  be the result of CCBI, and  $\mathcal{C}_{\Delta^{d,\Sigma}}$  be the result of the  $\Delta^{d,\Sigma}$ -combination of cases without domain knowledge ( $DK = \mathcal{U}$ ):

$$\mathcal{C}_{CCBI} = \bigcap_{i=1}^n \mathcal{B}_{h(d_{pb}(x_i, x_0))}^{\mathcal{U}_{so1}}(X_i)$$

$$\mathcal{C}_{\Delta^{d,\Sigma}} = \Phi_{x_0} \left( \mathcal{C}_{\Delta^{d,\Sigma}}^{\mathcal{U}}(SCS, Tgt) \right)$$

If CCBI provides a consistent result  $\mathcal{C}_{CCBI} \neq \emptyset$  then it coincides with the  $\Delta^{d,\Sigma}$ -case combination:

$$\mathcal{C}_{CCBI} = \mathcal{C}_{\Delta^{d,\Sigma}}$$

A proof for this proposition is given in appendix.

## 7 Computing IC Merging in Numerical Spaces

The computation of the merging  $\Delta_{IC}^{d,\Sigma}(M)$  is considered in this section with the assumptions:

- $\mathcal{U} = D_1 \times \dots \times D_n$  with  $D_i$  an interval of  $\mathbb{Z}$  or of  $\mathbb{R}$ .
- For  $x, y \in \mathcal{U}$   $d(x, y) = \sum_{i=1}^n w_i |y_i - x_i|$ .
- $IC$  can be defined by a finite set of linear inequalities.
- $M = \{A_1, \dots, A_p\}$  and every  $A_j$  can also be defined by a finite set of linear inequalities.

The computation of  $\Delta_{IC}^{d, \Sigma}(M)$  is equivalent to the minimization of the function  $y \mapsto d^\Sigma(M, y)$  under the constraint  $y \in IC$ .

**Proposition 3.** *The computation of  $\Delta_{IC}^{d, \Sigma}(M)$  is reducible to a linear programming problem [7].*

*Sketch of proof.* Minimizing  $d^\Sigma(M, y)$  is reducible to minimizing  $\sum_{j=1}^p \sum_{i=1}^n w_i |y_i - x_i^j|$  under the constraints  $x^j \in A_j$ . It is itself reducible to minimizing  $\sum_{j=1}^p \sum_{i=1}^n w_i z_i^j$  with the additional constraints:  $z_i^j \geq y_i - x_i^j$  and  $z_i^j \geq x_i^j - y_i$ , which is a linear programming problem.  $\square$

This property shows that if every  $D_i$  is an interval of  $\mathbb{R}$ , i.e. it is reducible to a real linear programming problem, then the computation cost is polynomial in  $n \times p$ . If any  $D_i$  is a subset of  $\mathbb{Z}$  it is a mixed integer linear programming (which is an NP-hard problem).

In particular, the running example of this paper has been computed this way (cf. section 5.3): the boolean dimensions are replaced by the integer interval  $[0, 1]_{\mathbb{Z}} = \{0, 1\}$ .

## 8 Conclusion, Related Work, and Future Work

The main contribution of this paper is to define an approach to case combination based on an IC belief merging operator. It can be applied to case adaptation as a particular case combination. It is shown to extend credible case-based inference. This approach is, *a priori*, applicable to any formalism on which a pre-IC merging operator can be defined, eg. a pseudo-metric space. Provided that cases are represented by numerical attributes and that the constraints and the distance are linear, belief merging can be reduced to linear programming. The complexity is then polynomial if there are only real value attributes (linear programming) and NP-hard otherwise (mixed integer linear programming). In propositional logic, IC merging and thus  $\Delta$ -combination is NP-hard, see [3].

Ongoing works are the implementation of a case combination based on an IC merging operator as defined in section 7 with the purpose of experimentation. The possibility to reduce other merging operators to linear programming should be investigated too.

As future work a systematic comparison with other case combination approaches should be performed. The convergence of these approaches should be investigated to determine in particular which ones can be covered by an IC merging operator. The following criteria –inspired by the case combination approaches review given in [8]– can

guide the comparison: how is structured the participation of each source case, whether the source cases are reused simultaneously or iteratively, and how the consistency is maintained.

Different ways of structuring the combination exist. Static structures as in Decentralized CBR (DzCBR) [9] where a set of contexts (or viewpoints) is set for the system. Every context generates a local solution according to its local domain knowledge and adaptation knowledge. Structure contained in cases as in DÉJÀ VU where the problem solving episodes are decomposed into a hierarchy of cases from the most abstract one that gives the main frame of the solution to the most concrete ones that solve sub-problems. Coverage of the target case by a set of source cases as in IDIOM [10] and COMPOSER [11] – a source case represent a partial solution with constraints for its inclusion in a global solution. In the approach presented in this paper all the source cases are equally considered, no explicit structure appears.

While COMPOSER, DzCBR, and the approach presented in this paper reuse the cases simultaneously, DÉJÀ VU and IDIOM do it iteratively. In DÉJÀ VU the resolution of a new query starts from the reuse of an abstract source case and is iterated on the resulting subproblems. In IDIOM a solution is built by iteratively incorporating source cases. A further investigation could be to investigate the possibility to express this approach to an iteration of conservative adaptation<sup>3</sup> and to relate it to a combination based on an IC merging operator.

Finally the approaches can be distinguished by the way the consistency is maintained. In DzCBR bridge rules between the contexts enforce the coherence of the local solutions altogether to form a global solution. IDIOM and COMPOSER use a conflict resolution algorithm. In our approach the inconsistencies between cases are managed by an IC merging operator. This motivates the investigation of relationships between conflict resolution and IC merging.

## References

1. Riesbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Inc., Hillsdale (1989)
2. Cojan, J., Lieber, J.: Conservative adaptation in metric spaces. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 135–149. Springer, Heidelberg (2008)
3. Konieczny, S., Lang, J., Marquis, P.: DA<sup>2</sup> merging operators. *Artificial Intelligence* 157(1-2), 49–79 (2004)
4. Hüllermeier, E.: Credible Case-Based Inference Using Similarity Profiles. *IEEE Transaction on Knowledge and Data Engineering* 19(6), 847–858 (2007)
5. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet functions for contraction and revision. *J. of Symbolic Logic* 50, 510–530 (1985)
6. Katsuno, H., Mendelzon, A.: Propositional knowledge base revision and minimal change. *Artificial Intelligence* 52(3), 263–294 (1991)
7. Karmarkar, N.: A new polynomial-time algorithm for linear programming. *Combinatorica* 4(4), 373–396 (1984)

<sup>3</sup> I.e., first computing  $\text{SolvedTgt}_1 = \text{ctxt}(\text{Srce}_1) \dot{+} \text{ctxt}(\text{Tgt})$  and then iteratively  $\text{SolvedTgt}_{i+1} = \text{ctxt}(\text{Srce}_i) \dot{+} \text{ctxt}(\text{SolvedTgt}_i)$ .

8. Gebhardt, F., Voß, A., Gräther, W., Schmidt-Belz, B.: Reasoning with complex cases. Kluwer, Boston (1997)
9. d'Aquin, M., Lieber, J., Napoli, A.: Decentralized case-based reasoning for the semantic web. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 142–155. Springer, Heidelberg (2005)
10. Smith, I., Lottaz, C., Faltings, B.: Spatial composition using cases: IDIOM. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 88–97. Springer, Heidelberg (1995)
11. Purvis, L., Pu, P.: An Approach to Case Combination. In: Voß, A. (ed.) Proc. of the ECAI 1996 Workshop: Adaptation in Case-Based Reasoning, pp. 43–46 (1996)

## Appendix

### Proof of Property 1

( $\Delta$ -1) The satisfaction of ( $\Delta$ -1) is straightforward from the definition of  $\Delta_{IC}^{d,\Sigma}(M)$ .

( $\Delta$ -3) If  $\bigcap M \cap IC \neq \emptyset$ , let  $y$  be an element of  $\bigcap M \cap IC$ ,  $y \in A$  for any  $A \in M$ , thus  $d(A, y) = 0$  and  $d^\Sigma(M, y) = 0$ . So  $d^\Sigma(M, IC) = 0$  and  $y \in \Delta_{IC}^{d,\Sigma}(M)$ .

On the other way round, if  $y \in \Delta_{IC}^{d,\Sigma}(M)$ , then  $d^\Sigma(M, y) = 0$  and  $d(A, y) = 0$  for any  $A \in M$  ( $d(A, y) \geq 0$  for any  $A$ ). The discretion assumption of  $(\mathcal{U}, d)$  implies then that  $y \in A$ . Indeed, consider the set  $\{x \in A \mid d(x, y) \leq 1\} \subseteq \mathcal{B}_1^{\mathcal{U}}(y)$ , it is finite and since  $d(A, y) < 1$  it is not empty, thus  $d(A, y)$  is reached for an  $x \in A$ .  $d(x, y) = 0$  which implies by the separation assumption that  $y \in A$ . Finally,  $y \in \bigcap M$  and by definition of  $\Delta_{IC}^{d,\Sigma}(M)$   $y \in IC$  which entails the result.

( $\Delta$ -5) and ( $\Delta$ -6) Their satisfaction is obvious when  $\Delta_{IC}^{d,\Sigma}(M_1) \cap \Delta_{IC}^{d,\Sigma}(M_2) = \emptyset$ . In the contrary, let  $y$  be an element of  $\Delta_{IC}^{d,\Sigma}(M_1) \cap \Delta_{IC}^{d,\Sigma}(M_2)$ ,

$$\begin{aligned} d^\Sigma(M_1, IC) + d^\Sigma(M_2, IC) &= d^\Sigma(M_1, y) + d^\Sigma(M_2, y) \\ &\geq \inf_{z \in IC} (d^\Sigma(M_1, z) + d^\Sigma(M_2, z)) \geq d^\Sigma(M_1 \cup M_2, IC) \end{aligned}$$

However

$$\begin{aligned} d^\Sigma(M_1, IC) + d^\Sigma(M_2, IC) &= \inf_{z \in IC} d^\Sigma(M_1, z) + \inf_{z \in IC} d^\Sigma(M_2, z) \\ &\leq \inf_{z \in IC} (d^\Sigma(M_1, z) + d^\Sigma(M_2, z)) \leq d^\Sigma(M_1 \cup M_2, IC) \end{aligned}$$

Thus all the inequalities can be replace by equalities, in particular the lower bound of  $d^\Sigma(M_1, z) + d^\Sigma(M_2, z)$  for  $z \in IC$  is  $d^\Sigma(M_1, y) + d^\Sigma(M_2, y) = d^\Sigma(M_1 \cup M_2, IC)$ , and as for  $i = 1, 2$   $d^\Sigma(M_i, z) \geq d^\Sigma(M_i, y)$  this lower bound is reached when  $d^\Sigma(M_i, z) = d^\Sigma(M_i, y) = d^\Sigma(M_i, IC)$ , ie.  $z \in \Delta_{IC}^{d,\Sigma}(M_1 \cup M_2)$  iff  $z \in \Delta_{IC}^{d,\Sigma}(M_1) \cap \Delta_{IC}^{d,\Sigma}(M_2)$ .

( $\Delta$ -7) and ( $\Delta$ -8) Similarly, if  $\Delta_{IC_1}^{d,\Sigma}(M) \cap IC_2 = \emptyset$ , the result is obvious.

Otherwise, let  $y$  be an element of  $\Delta_{IC_1}^{d,\Sigma}(M) \cap IC_2$ .

$$\begin{aligned} d^\Sigma(M, IC_1) &= \inf_{z \in IC_1} d^\Sigma(M, z) \leq \inf_{z \in IC_1 \cap IC_2} d^\Sigma(M, z) = d^\Sigma(M, IC_1 \cap IC_2) \\ d^\Sigma(M, y) &= d^\Sigma(M, IC_1) \leq d^\Sigma(M, IC_1 \cap IC_2) \leq d^\Sigma(M, y) \end{aligned}$$

Thus  $d^\Sigma(M, IC_1) = d^\Sigma(M, IC_1 \cap IC_2)$  and  $\Delta_{IC_1}^{d, \Sigma}(M_1) \cap IC_2 = \Delta_{IC_1 \cap IC_2}^{d, \Sigma}(M_1)$ .  
 ( $\Delta$ -2') Assume  $IC \neq \emptyset$  and every set in  $M$  is bounded.

If  $M = \emptyset$  then  $\bigcap M = \mathcal{U}$  and according to postulate ( $\Delta$ -3),  $\Delta_{IC}^{d, \Sigma}(M) = IC \neq \emptyset$ .

If  $M \neq \emptyset$ , then there exists an  $A \in M$ ,  $A \neq \emptyset$  and is bounded so finite. Let  $r = d^\Sigma(M, IC) + 1$  and consider  $S = \{y \in IC \mid d^\Sigma(M, y) \leq r\}$ .  $S$  is finite, indeed  $S \subseteq \{y \in \mathcal{U} \mid d^\Sigma(A, y) \leq r\} = \bigcup_{a \in A} \mathcal{B}_r^{\mathcal{U}}(a)$  which is a finite disjunction of finite sets. Moreover  $S \neq \emptyset$  as  $IC \neq \emptyset$  and the lower bound of  $x \mapsto d^\Sigma(x, M)$  on  $IC$  and  $S$  are equal.  $S$  being finite this lower bound is reached and  $\Delta_{IC}^{d, \Sigma}(M) \neq \emptyset$ .

( $\Delta$ -4) Assume  $d$  is symmetrical and consider three sets  $IC$ ,  $A_1 \subseteq IC$  and  $A_2 \subseteq IC$ .  
 If  $\Delta_{IC}^{d, \Sigma}(\{A_1, A_2\}) \cap A_1 \neq \emptyset$ , let  $y_1$  be an element of this set.

$$d^\Sigma(\{A_1, A_2\}, IC) = d^\Sigma(\{A_1, A_2\}, y_1) = d(A_1, y_1) + d(A_2, y_1)$$

as  $y_1 \in A_1$ ,  $d(A_1, y_1) = 0$  thus  $d^\Sigma(\{A_1, A_2\}, IC) = d(A_2, y_1)$ . From the discretion assumption, as  $A_2 \neq \emptyset$  there is  $y_2 \in A_2$  such that  $d(y_1, y_2) = d(y_2, y_1) = d(A_2, y_1)$ .

$$d^\Sigma(\{A_1, A_2\}, IC) = d(y_1, y_2) \geq d(A_1, y_2) = d^\Sigma(\{A_1, A_2\}, y_2)$$

As  $y_2 \in IC$   $d^\Sigma(\{A_1, A_2\}, y_2) \geq d^\Sigma(\{A_1, A_2\}, IC)$  thus, there is equality and  $y_2 \in \Delta_{IC}^{d, \Sigma}(\{A_1, A_2\})$  which entails that  $\Delta_{IC}^{d, \Sigma}(\{A_1, A_2\}) \cap A_2 \neq \emptyset$ .

The other implication is symmetrical.  $\square$

## Proof of Property 2

For  $X \in \mathcal{C}_{\text{CCBI}}$ ,  $d_{\text{so1}}(X_i, X) \leq h(d_{\text{pb}}(x_i, x_0))$  for any  $1 \leq i \leq n$ , so

$$d((x_i, X_i), (x_0, X)) = h(d_{\text{pb}}(x_i, x_0)) \leq \min_{Y \in \mathcal{U}_{\text{so1}}} d((x_i, X_i), (x_0, Y))$$

and

$$\begin{aligned} d^\Sigma(\text{CB}, (x_0, X)) &= \sum_{i=1}^n d((x_i, X_i), (x_0, X)) \leq \sum_{i=1}^n \left( \min_{Y \in \mathcal{U}_{\text{so1}}} d((x_i, X_i), (x_0, Y)) \right) \\ &\leq \min_{Y \in \mathcal{U}_{\text{so1}}} \left( \sum_{i=1}^n d((x_i, X_i), (x_0, Y)) \right) = \min_{(x_0, Y) \in \mathcal{U}} d^\Sigma(\text{CB}, (x_0, Y)) \end{aligned}$$

Thus,  $(x_0, X) \in \text{CC}_{\Delta^d, \Sigma}^{\mathcal{U}}(\text{CB}, \text{Tgt})$  and  $X \in \mathcal{C}_{\Delta^d, \Sigma}$ , which shows that  $\mathcal{C}_{\text{CCBI}} \subseteq \mathcal{C}_{\Delta^d, \Sigma}$ .

Moreover, if  $\mathcal{C}_{\text{CCBI}} \neq \emptyset$ , (ie. there is such an  $X$ ), then  $\min_{Y \in \mathcal{U}_{\text{so1}}} d^\Sigma(\text{CB}, (x_0, Y)) \leq d^\Sigma(\text{CB}, (x_0, X))$  and the previous inequalities are equalities. Thus, if  $Y \in \mathcal{U}_{\text{so1}}$  minimizes  $d^\Sigma(\text{CB}, (x_0, Y))$ , then  $\sum_{i=1}^n d((x_i, X_i), (x_0, Y)) = \sum_{i=1}^n d((x_i, X_i), (x_0, X))$ . As for any  $1 \leq i \leq n$   $d((x_i, X_i), (x_0, Y)) \geq d((x_i, X_i), (x_0, X))$ , this means that

$$d((x_i, X_i), (x_0, Y)) = d((x_i, X_i), (x_0, X)) = h(d_{\text{pb}}(x_i, x_0))$$

ie.  $Y \in \mathcal{C}_{\text{CCBI}}$ , which shows that  $\mathcal{C}_{\Delta^d, \Sigma} \subseteq \mathcal{C}_{\text{CCBI}}$ .  $\square$

# Maintenance by a Committee of Experts: The MACE Approach to Case-Base Maintenance

Lisa Cummins and Derek Bridge

Department of Computer Science,  
University College Cork,  
Ireland  
{l.cummins,d.bridge}@cs.ucc.ie

**Abstract.** Case-base administrators face a choice of many maintenance algorithms. It is well-known that these algorithms have different biases that cause them to perform inconsistently over different datasets. In this paper, we demonstrate some of the biases of the most commonly-used maintenance algorithms. This motivates our new approach: maintenance by a committee of experts (MACE). We create composite algorithms that comprise more than one individual maintenance algorithm in the hope that the strengths of one algorithm will compensate for the weaknesses of another. In MACE, we combine algorithms in two ways: either we put them in sequence so that one runs after the other, or we allow them to run separately and then vote as to whether a case should be deleted or not. We define a grammar that describes how these composites are created. We perform experiments based on 27 diverse datasets. Our results show that the MACE approach allows us to define algorithms with different trade-offs between accuracy and the amount of deletion.

## 1 Introduction

In this paper we examine the most commonly-used case-base maintenance algorithms, and we present a new approach to maintenance, the MACE approach, which uses a committee of experts to make maintenance decisions [\[1\]](#).

Case-base maintenance has the goal of restoring a degree of efficiency to the retrieval step of the CBR cycle by removing cases from the case-base, while, at the same time, preserving, or even enhancing, the accuracy of the system. The most common case-base maintenance algorithms are listed in Table [1](#).

As the Table shows, there are two types of case-base maintenance algorithm: those that delete noisy (or harmful) cases, and those that delete redundant cases. Noise reduction algorithms improve solution quality by removing cases that are considered to have a negative effect on system accuracy. Redundancy reduction algorithms improve system efficiency by removing cases which do not contribute to case-base competence.

---

<sup>1</sup> This material is based upon work supported by the Science Foundation Ireland under Grant Number 05/RFP/CMS0019.



**Table 1.** Atomic case-base maintenance algorithms, and classic composites

Name used in this paper	Name in the literature	Description
<i>Atomic noise reduction algorithms</i>		
RENN	RENN	Repeated Edited Nearest Neighbour [15]
BBNR	BBNR	Blame-Based Noise Reduction [5]
<i>Atomic redundancy reduction algorithms</i>		
ICFR	—	Redundancy reduction phase of ICF [2]
RCR	—	Redundancy reduction phase of RC [10]
CRR	CRR	Conservative Redundancy Reduction [5]
<i>Classic composite algorithms</i>		
RENN→ICFR	ICF	Brighton & Mellish’s Iterative Case Filtering [2]
RENN→RCR	RC	McKenna & Smyth’s algorithm [10]
BBNR→CRR	CBE	Delany & Cunningham’s Case-Base Editing algorithm [5]

In practice, case-base maintenance algorithms are often composites, comprising a noise reduction phase followed by a redundancy reduction phase. For example, Brighton & Mellish’s Iterative Case Filtering (ICF) algorithm comprises a RENN noise-filtering phase followed by their redundancy reduction phase [2].

The individual components of these composites have not always been tested individually, nor have they been tested in combination with other of the algorithms. For example, how does ICF’s redundancy reduction phase perform if it is not preceded by RENN? How does it perform if it is preceded by BBNR instead of RENN? To answer questions like these, we need to separate the composites into their two constituent parts. This is why we have extracted the redundancy reduction phase of the composite algorithms, naming them in the Table, and treating them as separate algorithms. For example, ICFR is our designation for the redundancy reduction phase of ICF.

In the next section we will analyse these algorithms in greater detail. Section 3 presents the MACE approach to case-base maintenance. Section 4 presents our experimental methodology. Then Sections 5, 6, 7 and 8 present overall results, results concerning noise reduction algorithms, results concerning the effect of class boundary complexity, and results for the special case of spam, respectively.

## 2 Comparison of Existing Algorithms

### 2.1 Empirical Comparisons

It is well-known that the composite algorithms in Table 1 perform differently on different datasets (see, e.g., [2]). The results in Table 2 from our own implementations of these algorithms exemplify this.<sup>2</sup> (Our experimental methodology is explained in detail later in this paper, Section 4.)

<sup>2</sup> Our implementations are publicly available as they are part of the open source jColibri framework, <http://gaia.fdi.ucm.es/projects/jcolibri/>

**Table 2.** Results for existing algorithms with highest results highlighted

Algorithm	27 datasets		Breathalyser		Credit		Lenses	
	Del (%)	Acc (%)	Del (%)	Acc (%)	Del (%)	Acc (%)	Del (%)	Acc (%)
RENN→ICFR	78.76	73.58	77.53	<b>74.00</b>	84.02	83.38	44.38	52.50
RENN→RCR	<b>88.75</b>	75.09	<b>87.66</b>	66.80	<b>87.84</b>	<b>86.38</b>	<b>86.25</b>	55.00
BBNR→CRR	55.31	<b>77.67</b>	61.95	71.20	55.90	83.62	68.13	<b>65.00</b>

If we consider the results averaged over 27 datasets, we see that RENN→RCR is the most aggressive and BBNR→CRR is the most conservative. Perhaps surprisingly, RENN→RCR is only slightly behind BBNR→CRR in accuracy even though it deletes over 30% more. Also, even though RENN→ICFR deletes 10% less than RENN→RCR, it is less accurate. However, since these results are averaged over 27 datasets, they hide details about individual datasets. For example, although RENN→RCR beats RENN→ICFR in the average results, this is not necessarily the case for each dataset.

If we look at the results from some of the individual datasets we see that RENN→ICFR has highest accuracy on the Breathalyser dataset, RENN→RCR on the Credit dataset, and BBNR→CRR on the Lenses dataset. On all three datasets, RENN→RCR deletes most. RENN→ICFR deletes more than BBNR→CRR on two of the datasets but the reverse is true for the Lenses dataset.

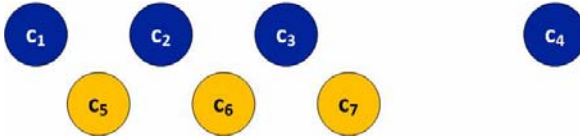
All of this simply serves to confirm what is well-known: in case-base maintenance, there is no clear ‘winning’ algorithm. The differences in performance of the algorithms are caused by their having different biases.

## 2.2 An Analysis of Algorithm Biases

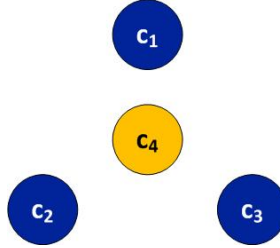
Each of the different atomic maintenance algorithms targets different types of cases to remove. In this section, we attempt to further illustrate these biases by looking at some scenarios in detail.

The two atomic noise reduction algorithms define noise differently. RENN regards a case as noisy if it has a different class to the majority of its  $k$  nearest neighbours. After each case has been checked, RENN deletes all cases that are flagged as noisy. This is then repeated until no more cases are removed [15]. BBNR identifies and removes cases which cause other cases to be misclassified [5]. It first classifies each case in the case-base using its neighbours. It removes neighbours that cause misclassifications, provided their removal does not cause cases that were previously correctly classified to become misclassified. Figures 1 and 2 illustrate scenarios which reveal biases in RENN and BBNR.

If we take  $k = 3$ , in the situation shown in Figure 1, each case has one nearest neighbour of the same class as itself and two of the other class. This means that the majority of the nearest neighbours of each case are of a different class and so RENN will flag each for deletion, leaving this case-base empty after the first deletion pass is made. This problem does not occur with BBNR because  $c_4$  does



**Fig. 1.** RENN problem situation



**Fig. 2.** BBNR problem situation

not cause any misclassification and so it will not be considered for deletion. It will be left as the only case in the case-base after BBNR is run.

The bias is the opposite in the scenario shown in Figure 2. Here, if we take  $k = 3$ , each of  $c_1$ ,  $c_2$  and  $c_3$  contributes to the misclassification of  $c_4$ , but if any of them is deleted then the others would be misclassified, so BBNR will delete nothing. RENN will only delete  $c_4$  because it is the only case with the majority of its  $k$  nearest neighbours of a different class.

In the case of the atomic redundancy reduction algorithms, all aim to remove cases that do not contribute to coverage, but they differ in how they decide which cases contribute. As a result they delete different types of cases. ICFR and CRR both aim to retain cases on the boundaries between classes because these cases are important for classification accuracy. ICFR removes cases that are solved by more cases than they themselves solve [2]. CRR removes cases that solve other cases [5]. It arranges the cases in the case-base in ascending order of how many cases they solve. It then adds each case  $c$  to a new case-base and removes from the original case-base any cases that  $c$  solves. RCR, however, aims to retain a case if it is surrounded by many cases of the same class, while deleting those that surround it [10]. It orders cases by descending relative coverage, which is a measure of the coverage contribution of each case in relation to how much it itself is covered. As each case is added to a new case-base, the cases that it solves are removed from the original case-base. Given these different biases, RCR will typically delete more cases than either ICF or CRR but this more aggressive deletion may result in lower accuracy in datasets with complex class boundaries.

It is apparent from both our experimental results and our brief analysis of biases that it is not the case that ‘one size fits all’ in case-base maintenance. The

$$\begin{aligned}
\langle System \rangle & ::= \langle AtomicAlgorithm \rangle \mid \langle Sequence \rangle \mid \langle Committee \rangle \\
\langle AtomicAlgorithm \rangle & ::= \langle AtomicNoiseReductionAlgorithm \rangle \mid \\
& \quad \langle AtomicRedundancyReductionAlgorithm \rangle \\
\langle AtomicNoiseReductionAlgorithm \rangle & ::= \text{'RENN'} \mid \text{'BBNR'} \\
\langle AtomicRedundancyReductionAlgorithm \rangle & ::= \text{'ICFR'} \mid \text{'RCR'} \mid \text{'CRR'} \\
\langle Sequence \rangle & ::= \langle System \rangle \text{'\(\rightarrow\} \langle System \rangle \\
\langle Committee \rangle & ::= \text{'\{'} \langle System \rangle \langle System \rangle \langle System \rangle^* \text{'\}'}, \langle VotingRule \rangle \\
\langle VotingRule \rangle & ::= \text{'S'} \mid \text{'M'} \mid \text{'U'}
\end{aligned}$$

**Fig. 3.** The MACE approach, defined by an EBNF grammar

question naturally arises whether novel composites, that combine algorithms with different biases, can do better than the atomic algorithms and the classic composites. We explore this in the next section, in which we present the MACE approach, where maintenance is done by a committee of experts.

### 3 Maintenance by a Committee of Experts (MACE)

In the MACE approach, we consider each atomic algorithm to be an expert that recommends cases for deletion. The MACE approach then defines different ways in which these atomic algorithms combine to form novel composite case-base maintenance algorithms. The easiest way to show how MACE forms these composites is by giving a grammar. This grammar is shown in EBNF in Figure 3.

The grammar's start symbol is  $\langle System \rangle$ , which has three expansions:

- $\langle System \rangle ::= \langle AtomicAlgorithm \rangle$  In the simplest case, a maintenance system comprises just one of the atomic algorithms. The atomic algorithms are here divided into the noise reduction algorithms (RENN, BBNR), and the redundancy reduction algorithms (ICFR, RCR, CRR).
- $\langle System \rangle ::= \langle Sequence \rangle$  A composite maintenance system may put systems into sequence, denoted by writing an arrow between them. An example is a classic composite such as BBNR $\rightarrow$ CRR. When we write this, we mean that the algorithm comprises two phases, where the second (CRR) is executed after the first (BBNR). This rule allows us to create all of the classic composites but novel composites that have not previously been tested too, such as ICFR $\rightarrow$ BBNR. (The recursion in the grammar also allows the possibility of sequences that comprise more than two algorithms, although this is a degree of freedom that we shall not explore in this paper.)
- $\langle System \rangle ::= \langle Committee \rangle$  Another way to create a composite is to form a committee (from which the MACE approach takes its name). A committee comprises a set of two or more systems, which we write between curly braces. Each system within a committee is executed, but cases are not deleted. If a member of a committee would ordinarily delete a case, we treat this instead as a vote for the deletion of that case. Committees must therefore also have

a voting rule, which we write in superscript following the closing curly brace, which determines how votes are tallied. We explain the voting rules in the next paragraph. An example of a committee is  $\{\text{BBNR}, \text{RENN}\}^U$  where each of BBNR and RENN separately proposes a set of cases to delete; and the committee deletes each case for which the voting is unanimous (U).

We define three voting rules for committees:

- Single (S):** If any member of the committee votes to delete case  $c$ , then the committee deletes  $c$ .
- Majority (M):** If more than half of the members of the committee vote to delete  $c$ , then the committee deletes  $c$ .
- Unanimous (U):** If all of the members of the committee vote to delete  $c$ , then the committee deletes  $c$ .

Single voting allows us to define committees that can aggressively delete large parts of a case-base, especially if the committees' constituents have very different biases. For example,  $\{\text{BBNR}, \text{RENN}\}^S$  deletes all the cases that BBNR identifies as noisy, plus all the cases that RENN identifies as noisy. On the other hand, with unanimous voting, we can define very conservative committees. For example, if  $\{\text{BBNR}, \text{RENN}\}^U$  deletes a case, we can be fairly confident that the case is noisy since both algorithms agree.

The real power of the grammar, however, comes from the mutual recursion in the rules. We will illustrate this with just three examples.

Since a sequence comprises two systems, and a system can be a committee, the grammar allows for sequences of committees. An example is  $\{\text{BBNR}, \text{RENN}\}^S \rightarrow \{\text{ICFR}, \text{RCR}, \text{CRR}\}^U$ , which first runs an aggressive noise reduction committee, followed by a very conservative redundancy reduction committee.

Similarly, since a committee comprises two or more systems, and a system can be a sequence, the grammar allows for committees of sequences. An example is  $\{\text{RENN} \rightarrow \text{ICFR}, \text{RENN} \rightarrow \text{RCR}, \text{BBNR} \rightarrow \text{CRR}\}^M$ , which uses majority voting of the three classic algorithms.

Finally, since a committee comprises two or more systems, and a system can be another committee, the grammar allows for committees with sub-committees. An example is  $\{\{\text{RENN}, \text{BBNR}\}^U, \text{RCR}\}^S$ , in which RCR votes alongside a noise sub-committee.

**Related Work.** The idea of combining techniques with different biases is not new. In machine learning, ensembles classify new problems using each of the classifiers in the ensemble [6,14]. Ensembles have similarly been used in CBR [4,9,12]. Similarly, distributed CBR [13] deals with the use of multiple case-bases and how the system works in combining these.

The work in case-based ensembles and in distributed CBR tends to imply the use of multiple case-bases, with goals such as improved accuracy, efficiency, or personalisation. Brodley & Friedl also use multiple case-bases (in fact, multiple folds of the same case-base) in their approach to case-base maintenance [3]. They split the case-base and use a classifier that is trained on one part to classify the

**Table 3.** Details of the datasets used in experiments

Dataset Name	Cases	Features	Classes	Accuracy (%)
Balance	625	4	3	85.12
Breast Cancer Diagnostic	569	30	2	96.90
Breast Cancer Prognostic	198	33	2	71.58
Breathalyser	127	5	2	71.60
Credit Approval	690	15	2	86.92
Dermatology	366	34	6	97.75
Flags	194	28	8	52.89
Glass Identification	214	9	7	69.05
Haberman’s Survival	306	3	2	69.51
Heart Disease Cleveland	303	14	5	53.22
Hepatitis	155	19	2	80.63
Ionosphere	351	33	2	86.71
Iris	150	4	3	97.00
Lenses	24	4	3	72.50
Lettings	756	5	2	84.97
Liver Disorders	345	6	2	64.20
Lung Cancer	32	56	3	48.00
Pima Indians Diabetes	768	8	2	70.78
Post-Operative Patient	90	8	3	64.71
Spam (5 datasets)	1000	700	2	95.55
Teaching Assistant Evaluation	151	5	3	55.33
Wine	178	13	3	96.67
Zoo	101	16	7	91.50
Average over 27 datasets	-	-	-	79.46

remaining part; they repeat this for each of several splits; and they then remove any cases that were misclassified. This use of multiple case-bases, however, is not a feature of the MACE approach: we are combining different *algorithms*. Closest to our work is arguably Wiratunga et al [17]. We are using a committee of maintenance experts, whereas they use a committee of adaptation experts.

## 4 Experimental Methodology

**Datasets.** Table 3 lists the 27 datasets that we use to evaluate maintenance algorithms in this paper: 20 from the UCI repository [1]; the Breathalyser dataset [7]; the Lettings dataset [11]; and five email datasets [5]. We have datasets of varying sizes, with different numbers of attributes and different numbers of classes. The datasets also have varying amounts of noise and redundancy.

**MACE Algorithms.** The MACE grammar defines an infinite set of maintenance algorithms. In our experiments, we put a number of restrictions on the sequences and committees that we created. We limited the sequences to ones that comprise either two atomic algorithms or one atomic algorithm and one

committee, and we obviously ensured that a sequence contained distinct algorithms (e.g. BBNR→BBNR is excluded). We similarly excluded duplicates from committees, and kept the length to five or less. We allowed sub-committees, but not sub-sub-committees, and a committee that contained a sub-committee could only contain one other component (which was allowed to be an atomic algorithm, a sequence or a sub-committee). With all of these restrictions in place, we created 307 algorithms from the grammar for experimentation.

**Methodology.** For evaluation, we performed repeated holdout on each of the datasets. Each dataset was divided randomly into three splits: a 60% training set, a 20% test set, and a final 20% which was required for evaluation of other systems in our research (not reported in this paper) and hence was discarded here. We created 10 different splits of the data.

We ran each algorithm on the training set and recorded the percentage of cases deleted. We also recorded the accuracy of the resulting case-base by using the test set as queries and recording the percentage correctly classified. We also recorded the accuracy before performing any maintenance to provide a benchmark figure. Table 3 contains these benchmark accuracies.

## 5 General Results

In this section, we compare overall performance, averaged over the 27 datasets. But there is an immediate problem: case-base maintenance is a multi-objective problem. We wish to optimise both the percentage of cases deleted, but also the accuracy of the final case-base. This is not possible: algorithms that do well on one of the criteria do not necessarily do well on the other. It comes as no surprise, for example, that our experimental results show that committees with many members and single voting delete many cases, but at a severe cost in accuracy; the opposite is the case with committees with few members and unanimous voting. Case-base administrators must strike a balance between accuracy and deletion. They need ways of seeing the trade-offs, so they can make informed decisions. We looked at two ways of presenting this.

### 5.1 Harmonic Mean

We could present the arithmetic mean of the percentage of cases deleted and the case-base accuracy. But this can be misleading. The arithmetic mean in the case of an algorithm with very high accuracy and very low deletion will be similar to the arithmetic mean in the case of an algorithm with medium accuracy and deletion. To avoid this, we instead use the harmonic mean (Equation 1). The harmonic mean penalises large differences between two values so that a high mean is only produced if both individual values are high. In this way we can find algorithms which have a high value for both accuracy and deletion.

$$\text{HarmonicMean}(\text{Acc}, \text{Del}) = \frac{2 \times \text{Acc} \times \text{Del}}{\text{Acc} + \text{Del}} \quad (1)$$

**Table 4.** Top five algorithms ordered by harmonic mean over 27 different datasets

Algorithm	Deletion (%)	Accuracy (%)	Harmonic mean
$\{\text{RENN, BBNR, RCR}\}^S$	90.12	74.94	81.83
RENN $\rightarrow$ RCR	88.75	75.09	81.35
$\{\text{RENN, RCR}\}^S$	88.20	74.15	80.57
RCR $\rightarrow$ BBNR	89.67	72.63	80.26
$\{\{\text{RENN, BBNR}\}^U, \text{RCR}\}^S$	83.26	77.42	80.23

Table 4 shows the algorithms with the top five harmonic means. We can see that these algorithms have high values for both accuracy and deletion.

These algorithms perform well when compared with the average accuracy when no deletion occurs (79.46%). We can see that the algorithm with the best accuracy,  $\{\{\text{RENN, BBNR}\}^U, \text{RCR}\}^S$ , only causes a 2% drop in accuracy while deleting 83.26%. This seems to be a reasonable compromise.

Additionally, we can see that novel MACE algorithms are performing well. In the top five, there are three committees, all of which are quite conservative. There is also one novel sequence, RCR $\rightarrow$ BBNR, and one classic, RENN $\rightarrow$ RCR. Interestingly, all five are some combination of RCR and a noise reduction algorithm. On its own, the atomic RCR algorithm has much lower deletion (76.67%) and therefore a much lower harmonic mean value (76.53). Our results show the strength of combining algorithms in sequences and committees: the weaknesses in RCR that cause the accuracy drop are compensated for by the noise algorithms.

A problem with this way of presenting the results, however, is that it does not give the case-base administrator much sense of what trade-offs can be made. For example, what does she do if she is not happy to see accuracy fall by 2%.

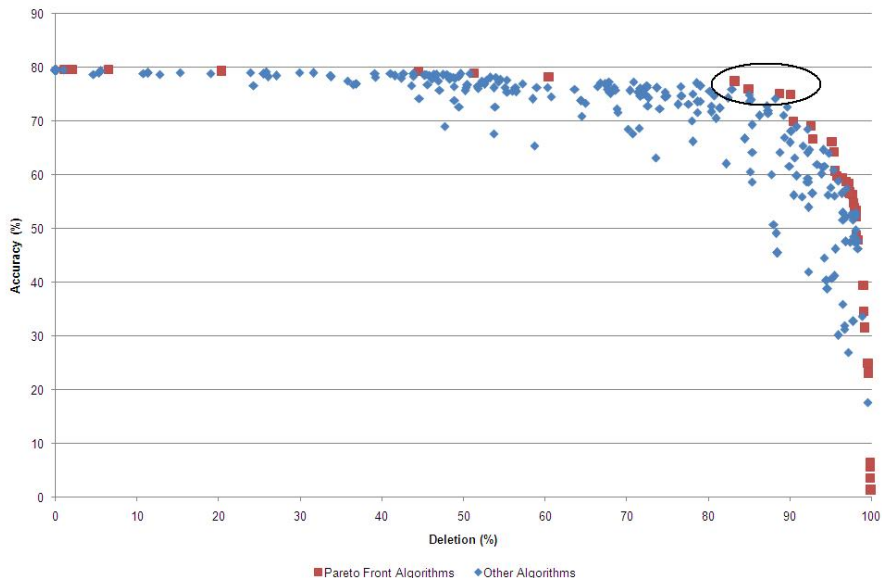
## 5.2 Pareto Front

Another way to handle a multi-objective problem is to compute the Pareto front. The Pareto front contains algorithms that are not dominated by any other algorithms. We take one maintenance algorithm to dominate another if and only if it both deletes more and is more accurate. This finds a set of algorithms which are not bettered by other algorithms and which are all either equal to one another or incomparable with one another (e.g. because one of them deletes more, but the other has higher accuracy).

In Figure 4, we plot all 307 of our algorithms. The percentage of cases deleted is on the  $x$ -axis, and the percentage accuracy is on the  $y$ -axis. Each point represents one algorithm: red squares are algorithms on the Pareto front; blue diamonds are algorithms that are dominated by those in the Pareto front.

The Pareto front in Figure 4 contains algorithms that delete everything and hence have extremely low accuracy, and vice versa. However, it also contains algorithms with a good balance between the two. These are very easy to identify on the graph because we can see where accuracy begins to fall rapidly. This is the point where 80% or more of the case-base is deleted. We have circled these





**Fig. 4.** Result on 27 different datasets, highlighting the Pareto front

algorithms on the graph. The four algorithms we have circled are:  $\{\{\text{RENN}, \text{BBNR}\}^U, \text{RCR}\}^S$ ,  $\text{BBNR} \rightarrow \text{RCR}$ ,  $\text{RENN} \rightarrow \text{RCR}$  and  $\{\text{RENN}, \text{BBNR}, \text{RCR}\}^S$ . Again, we can see that sequences and committees containing RCR and the noise algorithms perform strongly.

The advantage of the graph is that a case-base administrator can investigate the compromises that need to be made. For example, if she wants to delete 50% of the case-base, the graph shows that this can be done while retaining 78% accuracy; but if she wants to delete 90%, then accuracy drops to 75%. Similarly, if she wants to keep accuracy above 79%, the most that she can delete is about 45%. Of course, to be truly useful, the administrator needs a graph that shows the Pareto front for her particular case-base, not an average over 27 datasets.

## 6 Noise-Filtering

The idea of using an initial noise reduction phase followed by a redundancy reduction phase probably originates with Wilson et al [16]. All three classic case-base maintenance algorithms follow suit. Here, inspired by the numerous alternatives that the MACE approach defines, we wanted to determine how beneficial a noise-filtering phase is. Accordingly, in Table 5, we compare all five atomic algorithms, all three classic composite algorithms, algorithms in which atomic redundancy reduction algorithms are paired with noise reduction algorithms that they have never previously been paired with (e.g.  $\text{BBNR} \rightarrow \text{RCR}$ ), and algorithms in which the noise reduction phase comes after the redundancy

**Table 5.** Results for different uses of noise reduction algorithms

Algorithm	27 datasets		Breathalyser		Lenses	
	Del (%)	Acc (%)	Del (%)	Acc (%)	Del (%)	Acc (%)
RENN	24.27	76.57	25.84	68.80	36.25	52.50
BBNR	23.85	<b>79.06</b>	25.97	71.60	41.25	60.00
ICFR	60.77	74.60	61.30	66.00	40.00	67.50
RCR	76.67	76.40	77.40	70.80	63.75	<b>72.50</b>
CRR	35.76	77.51	41.17	72.00	32.50	<b>72.50</b>
RENN→ICFR	78.76	73.58	77.53	<b>74.00</b>	44.38	52.50
RENN→RCR	88.75	75.09	87.66	66.80	86.25	55.00
RENN→CRR	60.27	76.23	66.62	67.20	71.25	55.00
BBNR→ICFR	74.95	74.69	72.08	67.20	70.00	55.00
BBNR→RCR	84.99	75.90	85.19	66.80	80.00	65.00
BBNR→CRR	55.31	77.67	61.95	71.20	68.13	65.00
ICFR→RENN	85.43	64.20	87.79	50.80	79.38	45.00
RCR→RENN	<b>91.65</b>	65.46	<b>92.60</b>	52.00	<b>92.50</b>	45.00
CRR→RENN	64.95	73.35	72.73	62.40	75.63	47.50
ICFR→BBNR	81.47	72.39	80.52	59.20	76.88	57.50
RCR→BBNR	89.67	72.63	87.79	69.20	91.88	45.00
CRR→BBNR	60.46	78.20	64.68	70.00	75.63	62.50

reduction phase, rather than before it (e.g. RCR→RENN). In addition to averages over all 27 datasets, we look separately at the Breathalyser dataset, which is known to be very noisy (since it was collected in Dublin pubs!), and the Lenses dataset, which is known to be noise-free. As Table 3 shows, the accuracy of these datasets is 79.46%, 71.60% and 72.50%, respectively without maintenance.

Firstly we look at the atomic algorithms and their performance over the datasets. We see that the atomic *noise* reduction algorithms (RENN, BBNR) never actually improve accuracy, even on the noisy Breathalyser data (although BBNR does maintain the same accuracy here while deleting 25.97%). We also see that they both delete a large proportion of the Lenses dataset even though this is noise-free, and they both cause large accuracy drops as a result. Unexpectedly, all three atomic *redundancy* reduction algorithms have higher accuracy than the noise reduction algorithms on this dataset. In fact, CRR has a higher accuracy than RENN across the table, and only loses to BBNR on the results averaged over the 27 datasets. RCR has similar accuracy to both RENN and BBNR on the Breathalyser dataset while deleting over 50% more of the case-base.

Secondly, we look at the changes in accuracy and deletion when each of the noise reduction algorithms is run before each of the redundancy reduction algorithms. For each such algorithm, on the Lenses dataset, there is a large drop in accuracy. This is not surprising given the fact that the atomic noise reduction algorithms performed so badly on this dataset. For the Breathalyser dataset, only RENN→ICFR increases accuracy; BBNR→CRR has the smallest fall in accuracy and yet deletes the most. In the results averaged over the 27 datasets, changes in accuracy are quite small. Using BBNR in the noise reduction phase gives slightly less loss of accuracy than using RENN.

When we look at the deletion results, we can see that these composites do delete more than their constituents on their own, as we would expect. Mostly, deleting more cases lowers accuracy. But there are exceptions where accuracy improves (e.g. RENN→ICFR on the Breathalyser dataset) and where accuracy falls only a little while the case-base shrinks a lot (e.g. BBNR→CRR on the Breathalyser dataset). Interestingly, these good results come from ‘classic’ algorithms (RENN→ICFR, BBNR→CRR). But there are novel combinations that are doing well on the average results, e.g. RENN→CRR, which might be worthy of investigation on other individual datasets.

Finally, we look at the effect of switching around the sequences so that the redundancy reduction algorithm comes before the noise reduction algorithm. The results here are very consistent. In all situations, the resulting algorithm deletes a greater amount than it does in the conventional ordering. On the other hand, the resulting algorithm is always less accurate than its original with three exceptions: CRR→BBNR is more accurate than BBNR→CRR on the averaged data, RCR→BBNR is more accurate than BBNR→RCR on the Breathalyser data, and ICFR→BBNR is more accurate than BBNR→ICFR on the Lenses dataset. Of these, RCR→BBNR might be worthy of further investigation.

In summary, this analysis shows that the noise reduction phase that is used by so many case-base maintenance algorithms is not always useful, and in some cases may be quite detrimental to the accuracy of the algorithm. It also shows that the three classic composite algorithms do not necessarily use the best algorithm for their noise reduction phase, and that the best algorithm to use can change depending on the dataset.

## 7 The Effect of Boundary Complexity

The complexity of the boundary between classes in a dataset may have an effect on the performance of the different maintenance algorithms. For example, as noted previously, since RCR retains a case  $c$  if it is surrounded by cases of the same class as  $c$ , rather than retaining boundary cases, it may cause a loss of accuracy when boundaries are complex. To explore this idea, we computed a boundary complexity measure on the datasets. The boundary complexity measure that we use is the intra/inter class distance ratio [8]. We looked at the two datasets with highest complexity (Lettings and Flags) and the two with lowest complexity (Zoo and Iris) according to this measure. For each of these datasets, we found the top five algorithms ordered by harmonic mean. These top five algorithms are shown in Tables 6 and 7.

We can see that the top five algorithms for the datasets with highest complexity are quite similar. Almost all of them are committees of two sequences, where the sequences comprise a noise reduction algorithm along with either RCR or CRR. The top algorithms for the datasets with lowest complexity are also very similar to each other, with three algorithms common to both datasets. We can see that these algorithms are much more like the ones that did well overall (Section 5) containing one or both of the noise reduction algorithms along with RCR. We also note that the top algorithms for the complex datasets are quite

**Table 6.** Top five algorithms ordered by harmonic mean for most complex datasets

Lettings	Flags
$\{\text{BBNR} \rightarrow \text{CRR}, \text{RENN} \rightarrow \text{RCR}\}^S$	$\{\text{RENN} \rightarrow \text{CRR}, \text{CRR} \rightarrow \text{BBNR}\}^S$
$\{\text{RCR} \rightarrow \text{BBNR}, \text{CRR} \rightarrow \text{RENN}\}^S$	$\{\text{RENN}, \text{BBNR}, \text{CRR}\}^S$
$\{\text{CRR} \rightarrow \text{RENN}, \text{RCR} \rightarrow \text{BBNR}\}^S$	$\{\text{RENN} \rightarrow \text{RCR}, \text{RCR} \rightarrow \text{BBNR}\}^U$
$\{\text{RENN} \rightarrow \text{RCR}, \text{CRR} \rightarrow \text{BBNR}\}^S$	$\{\text{RENN} \rightarrow \text{CRR}, \text{RCR} \rightarrow \text{RENN}\}^U$
$\{\text{RCR} \rightarrow \text{RENN}, \text{RENN} \rightarrow \text{ICFR}\}^S$	$\{\text{CRR} \rightarrow \text{RENN}, \text{RCR} \rightarrow \text{BBNR}\}^U$

**Table 7.** Top five algorithms ordered by harmonic mean for least complex datasets

Zoo	Iris
$\text{RENN} \rightarrow \text{RCR}$	$\{\text{RENN}, \text{RCR}\}^S$
$\{\text{RENN}, \text{BBNR}, \text{RCR}\}^S$	$\{\text{RENN}, \text{BBNR}, \text{RCR}\}^S$
$\{\text{RENN} \rightarrow \text{RCR}, \text{RCR} \rightarrow \text{RENN}\}^U$	$\text{RENN} \rightarrow \text{RCR}$
$\{\{\text{RENN}, \text{BBNR}\}^U, \text{RCR}\}^S$	$\{\{\text{RENN}, \text{BBNR}\}^U, \text{RCR}\}^S$
$\text{RCR}$	$\text{BBNR} \rightarrow \text{RCR}$

different to the top algorithms for the simple datasets. Interestingly too, the classic  $\text{RENN} \rightarrow \text{RCR}$  algorithm is in the top three for both the Zoo and the Iris datasets. However, it comes in 16th place for the Flags dataset, and in 52nd place for the Lettings dataset. This suggests that there is a need for investigation of maintenance algorithms that are suited to datasets with complex boundaries.

## 8 The Special Case of Spam

Spam-filtering is a task with special characteristics [5]. Of particular relevance here are the facts that spam is heterogeneous (hence, it is a disjunctive concept), and there is a high cost of false positives. We decided, therefore, to look separately at how the maintenance algorithms perform on our five spam datasets.

As well as recording the percentages of accuracy and deletion for each of the algorithms, we also recorded the rate of false positives, the rate of false negatives and the within-class error rate (the average of the other two rates) [5]. Table 8 shows the best five algorithms, ordered by increasing within-class error rate.

**Table 8.** Top five algorithms ordered by within-class error rate for the spam datasets

Algorithm	Del	Acc	FP Rate	FN Rate	Err Rate
BBNR	5.84	96.16	2.64	5.02	3.82
$\{\{\text{CRR}, \text{RCR}, \text{ICFR}\}^U, \text{BBNR}\}^S$	35.58	95.86	2.42	5.84	4.14
$\{\text{RCR}, \text{BBNR}\}^U$	4.54	95.64	2.46	6.26	4.36
$\{\text{BBNR} \rightarrow \text{CRR}, \text{BBNR} \rightarrow \text{ICFR}, \text{ICFR} \rightarrow \text{BBNR}, \text{CRR} \rightarrow \text{BBNR}, \text{BBNR} \rightarrow \text{RCR}, \text{RCR} \rightarrow \text{BBNR}\}^U$	28.88	95.60	2.44	6.30	4.38
$\{\text{CRR} \rightarrow \text{BBNR}, \text{BBNR} \rightarrow \text{RCR}\}^U$	38.14	95.56	2.22	6.66	4.44

We can see that the algorithms that do well on the spam datasets are quite different from the algorithms that have done well on other datasets. BBNR performs very strongly, with the lowest rate of error overall. It also provides the noise removal component of all of the sequences and committees in the top five; RENN is not contained in any of the top five. Since BBNR was developed specifically to remove noise from spam datasets, this result is not surprising. However, it does confirm the strength of the algorithm for this domain.

It is also interesting to note that CRR, the algorithm developed specifically to remove redundancy from spam datasets, does not perform as strongly. It is contained in three of the top five algorithms, but appears less often than RCR. Also, the ‘classic’ BBNR→CRR composite comes only in 62nd place.

This indicates that, while the BBNR part of the composite algorithm is well suited to spam datasets, CRR is not as well suited. In fact, it appears that no single redundancy removal algorithm on its own deals well with the spam datasets. The committees in the top five all contain at least two atomic redundancy removal algorithms, if not all three. This may be due to the fact that spam is heterogeneous; it is more difficult to be sure that spam cases are redundant because they are distributed quite widely across the case base.

## 9 Conclusions and Future Work

In this paper we have investigated the most commonly-used case-base maintenance algorithms and have shown their strengths and weaknesses. We presented our MACE approach, which allows us to combine these algorithms. We investigated the performance of 307 algorithms defined by MACE using 27 datasets.

Our MACE algorithms performed strongly: four of the top five algorithms were new sequences or committees. As well as reporting the top algorithms over the 27 datasets, we looked at three particular areas where results could be different. We examined the initial noise reduction phase that the classic algorithms use and concluded that it is not always beneficial. We looked at boundary complexity and the effect that this has on the maintenance algorithms. We showed that the RCR algorithm, which works well on simple datasets, performs less well on those with complex boundaries. We also looked at the spam domain and showed that the BBNR algorithm does very well on this domain.

Our ongoing work consists of predicting a good maintenance algorithm for a given dataset based on properties of that dataset. In particular, we are investigating using a ‘meta-case-base’ for this task.

## References

1. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository (2007)
2. Brighton, H., Mellish, C.: On the Consistency of Information Filters for Lazy Learning Algorithms. In: Rauch, J., Zytkow, J.M. (eds.) PKDD 1999. LNCS, vol. 1704, pp. 283–288. Springer, Heidelberg (1999)
3. Brodley, C.E., Friedl, M.A.: Identifying and Eliminating Mislabeled Training Instances. In: AAAI/IAAI, pp. 799–805. AAAI Press, Menlo Park (1996)

4. Cunningham, P., Zenobi, G.: Case Representation Issues for Case-Based Reasoning from Ensemble Research. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS, vol. 2080, pp. 146–157. Springer, Heidelberg (2001)
5. Delany, S.J., Cunningham, P.: An Analysis of Case-Based Editing in a Spam Filtering System. In: Funk, P., González-Calero, P. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 128–141. Springer, Heidelberg (2004)
6. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
7. Doyle, D., Cunningham, P., Bridge, D.G., Rahman, Y.: Explanation Oriented Retrieval. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 157–168. Springer, Heidelberg (2004)
8. Ho, T.K., Basu, M.: Measuring the Complexity of Classification Problems. In: Proc. of the 15th International Conference on Pattern Recognition, 2000, pp. 43–47 (2000)
9. Leake, D.B., Sooriamurthi, R.: When Two Case Bases Are Better than One: Exploiting Multiple Case Bases. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS, vol. 2080, pp. 321–335. Springer, Heidelberg (2001)
10. McKenna, E., Smyth, B.: Competence-Guided Case-Base Editing Techniques. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS, vol. 1898, pp. 186–197. Springer, Heidelberg (2000)
11. Nicholson, R., Bridge, D., Wilson, N.: Decision Diagrams: Fast and Flexible Support for Case Retrieval and Recommendation. In: Roth-Berghofer, T., Göker, M.H., Altay Güvenir, H. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 136–150. Springer, Heidelberg (2006)
12. Orecchioni, A., Wiratunga, N., Massie, S., Craw, S.: k-NN Aggregation with a Stacked Email Representation. In: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 415–429. Springer, Heidelberg (2008)
13. Plaza, E., McGinty, L.: Distributed Case-Based Reasoning. *The Knowledge Engineering Review* 20(3), 261–265 (2006)
14. Quinlan, R.J.: Bagging, Boosting, and C4.5. In: AAAI/IAAI, vol. 1, pp. 725–730. AAAI Press, Menlo Park (1996)
15. Tomek, I.: An Experiment with the Edited Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics* 6(6), 448–452 (1976)
16. Wilson, D.R., Martinez, T.R.: Instance Pruning Techniques. In: Fisher, D. (ed.) Proc. of the 14th International Conference on Machine Learning, ICML, pp. 403–411. Morgan Kaufmann, San Francisco (1997)
17. Wiratunga, N., Craw, S., Rowe, R.: Learning to Adapt for Case-Based Design. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416, pp. 421–435. Springer, Heidelberg (2002)

# The Good, the Bad and the Incorrectly Classified: Profiling Cases for Case-Base Editing

Sarah Jane Delany

Dublin Institute of Technology, Dublin, Ireland  
sarahjane.delany@dit.ie

**Abstract.** Case-based approaches to classification, as instance-based learning techniques, have a particular reliance on training examples that other supervised learning techniques do not have. In this paper we present the RDCL case profiling technique that categorises each case in a case-base based on its classification by the case-base, the benefit it has and/or the damage it causes by its inclusion in the case-base. We show how these case profiles can identify the cases that should be removed from a case-base in order to improve generalisation accuracy and we show what aspects of existing noise reduction algorithms contribute to good performance and what do not.

## 1 Introduction

Unlike many other supervised learning techniques, lazy learning techniques are instance-based and depend greatly on individual training examples. This has motivated considerable research into the identification of appropriate training examples for case-based maintenance tasks. Case-base editing involves reducing a case-base or training set to a smaller number of cases while trying to maintain and even improve the generalisation accuracy. A key aspect of case-base editing is to identify and remove noisy or exceptional cases that can cause a degradation in the generalisation accuracy.

In this paper we present a technique for associating a competence profile with each case in a case-base. The case profile categorises each case based on three characteristics;

- (i) whether the case is classified correctly or not by the rest of case-base,
- (ii) what benefit (or good) if any, it brings to the case-base by its inclusion, and
- (iii) whether or not it causes damage (or harm) to the case-base by its inclusion.

Different combinations of the three characteristics result in a case having one of eight possible profiles. Building on established case-base maintenance research [12], these case profiles are derived from a competence model constructed on the case-base by a leave-one-out classification of all cases.

A key advantage of identifying the types of cases in a case-base is that it exposes the effect of removing cases of different types from a training set. This facilitates identifying which case types are the most useful in maintaining and improving generalisation accuracy.

A further important benefit of this profiling methodology is that it reveals the biases of existing noise reduction techniques and shows what aspects contribute to good performance and what aspects do not. Resulting from our analysis of these case profiles we present in this paper a simple noise reduction technique based on these profiles that consistently improves generalisation accuracy and we compare its performance to existing noise reduction techniques across a number of datasets.

The rest of this paper is structured as follows. Section 2 of this paper reviews the extensive existing case-editing literature. Section 3 presents our case profiling approach while section 4 describes the investigations we performed into showing which types of cases are most beneficial to remove from a case-base. This section also includes an investigation into what kinds of cases existing noise reduction algorithms remove and presents our profile-based noise reduction technique. Section 5 concludes with some directions for future work.

## 2 Case-Base Editing

Case-base editing techniques have been categorised as *competence preservation* or *competence enhancement* techniques [3,4]. Competence preservation corresponds to redundancy reduction, removing superfluous cases that do not contribute to classification competence. Competence enhancement is effectively noise reduction, removing noisy or corrupt cases from the training set. Competence preservation techniques aim to remove internal cases in a cluster of cases of the same class and can predispose towards preserving noisy cases as exceptions or border cases. Competence enhancement on the other hand aims to remove noisy or corrupt cases but can remove exceptional or border cases which may not be distinguishable from true noise, so a balance of both can be useful in a case editing algorithm.

Editing strategies normally operate in one of two ways; *incremental* which involves adding selected cases from the training set to an initially empty edited set and *decremental* which involves contracting the training set by removing selected cases.

An early competence preservation technique is Hart's Condensed Nearest Neighbour (CNN) [5]. CNN is an incremental technique which adds to an initially empty edited set any case from the training set that cannot be classified correctly by the edited set. This technique is very sensitive to noise and to the order of presentation of the training set cases, in fact CNN by definition will tend to preserve noisy cases. Ritter [6] reported improvements on the CNN with their Selective Nearest Neighbour (SNN) which imposes the rule that every case in the training set must be closer to a case of the same class in the edited set than to any other training case of a different class. Gates [7] introduced a decremental technique which starts with the edited set equal to the training set and removes a case from the edited set where its removal does not cause any other training case to be misclassified. This technique will allow for the removal of noisy cases but is sensitive to the order of presentation of cases. More recent improvements to CNN have been proposed by Chou et al. [8] and Angiulli [9] with Hao et al. [10] proposing a variation appropriate for text classification.



Competence enhancement or noise reduction techniques start with Wilson's Edited Nearest Neighbour (ENN) algorithm [11], a decremental strategy, which removes cases from the training set which do not agree with their  $k$  nearest neighbours. These cases are considered to be noise and appear as exceptional cases in a group of cases of the same class.

Tomek [12] extended this with his repeated ENN (RENN) and his *all k-NN* algorithms. Both make multiple passes over the training set, RENN repeating the ENN algorithm until no further eliminations can be made from the training set while all  $k$ -NN uses incrementing values of  $k$  for each case and removes the case if a misclassification occurs for any value of  $k$ . These techniques focus on noisy or exceptional cases and do not result in the same storage reduction gains as the competence preservation approaches. A variation on ENN using  $k$  nearest centroid neighbours instead of  $k$  nearest neighbours was proposed by Sánchez et al. [13]. There is also work which considered relabelling examples rather than deleting them [14,15].

Later editing techniques can be classified as hybrid techniques incorporating both competence preservation and competence enhancement stages. Aha [16] presented a series of Instance Based (IB) learning algorithms to reduce storage requirements and tolerate noisy instances. IB2 is similar to CNN adding only cases that cannot be classified correctly by the reduced training set. IB2's susceptibility to noise is handled by IB3 which records how well cases are classifying and only keeps those that classify correctly to a statistically significant degree. Other researchers have provided variations on the  $IB_n$  algorithms [17,18,19].

More recently, Pan et al. [20] proposed a case-base mining algorithm Kernel-based Greedy Case-base Mining (KGCM) guided by theoretical results to edit a case-base. It involves using a kernel transformation to map the original case-base to a new feature space and using FDA to help remove noise and identify the predictive features. KGCM is an incremental approach that considers cases from this new space for addition based on their diversity.

Also, Massie et al.'s [21] recent work on case-base profiling introduces a decremental noise reduction strategy called Threshold Error Reduction (TER) that removes cases based on a complexity measure called the Friend:Enemy (F:E) Ratio. This measure compares the distances to a case's nearest like neighbours with distances to its nearest unlike neighbours. Their case-editing algorithm iteratively removes cases with F:E ratios higher than certain thresholds to remove noisy cases and to smooth out the boundary. They point out that different datasets are threshold dependent.

## 2.1 Competence-Based Case-Base Editing

More recent approaches to case-base editing build a competence model of the training data and use the competence properties of the cases to determine which cases to include in the edited set. Measuring and using case competence to guide case-base maintenance was first introduced by Smyth & Keane [1]. They introduced two important competence properties, the *reachability* and *coverage* sets for a case in a case-base [1]. The *reachability set* of a case  $c$  is the set of

all cases that can successfully classify  $c$ , and the *coverage set* of a case  $c$  is the set of all cases that  $c$  can successfully classify. The coverage and reachability sets represent the local competence characteristics of a case and are used as the basis of a number of editing techniques. Smyth & Keane first used these case competence properties in their Footprint Deletion policy which identified a series of case categories using these competence properties to provide a means of ordering cases for deletion. The *competence footprint* is a subset of the case-base that provides the same competence as the entire case-base.

McKenna & Smyth [22] later presented a family of competence-guided editing methods for case-bases which combine both incremental and decremental strategies. This family of algorithms is based on different combinations of policies for adding and removing cases, policies for presenting cases for consideration and for competence model update. These algorithms also include an RENN based initial pass to remove noise. Brighton & Mellish [3] also use the coverage and reachability properties of cases in their Iterative Case Filtering (ICF) algorithm. ICF is a decremental strategy contracting the training set by removing those cases  $c$ , where the number of other cases that can correctly classify  $c$  is higher than the number of cases that  $c$  can correctly classify. This strategy focuses on removing cases far from class borders. ICF also includes a pre-processing noise reduction stage, effectively RENN, to remove noisy cases.

Wilson & Martinez [23] presented a series of Reduction Technique (RT) algorithms which they later enhanced into the Decremental Reduction Optimisation Procedures (DROP) [4]. Although these were originally published before the definitions of coverage and reachability, they could also be considered to use a competence model. They define the set of *associates* of a case  $c$  which is comparable to the coverage set of McKenna and Smyth except that the associates set will include cases of a different class from case  $c$  whereas the coverage set will only include cases of the same class as  $c$ . The  $RT_n$  and  $DROP_n$  algorithms use a decremental strategy.

In contrast to the earlier approaches to noise reduction which tend to focus on removing the cases that are misclassified, Delany & Cunningham's Blame Based Noise Reduction (BBNR) [2] attempts to identify those cases causing the misclassifications and uses this information to identify training cases the case-base would be better off without. BBNR extends Smyth & Keane's case competence model by including an additional set, the *liability set* which is the set of all cases that  $c$  causes to be misclassified. This attempts to model the situation of a case being classified incorrectly because of the retrieved cases that contributed to its classification rather than the case being itself a noisy or mislabelled case.

### 3 Case Profiles

In this section we propose an approach to modelling the competence of a case-base with a view to categorising the competence of each case in the case-base. We then have the opportunity to investigate the effect that each type of case can have on case-base competence.

### 3.1 Enhanced Competence Model

Smyth & Keane's [1] case-base competence modelling approach proposed two sets to model the local competence properties of a case, the *reachability set* of a case  $c$ , the set of all cases that can successfully classify  $c$  and the *coverage set* of a case  $c$ , the set of all cases that  $c$  can classify. Using the case-base itself as a representative of the target problem space, these sets can be estimated as shown in Equations 1 and 2. Delany & Cunningham's [2] extension included an additional property; the *liability set* of a case  $c$ , the set of all cases that  $c$  causes to be misclassified and can be estimated by Equation 3.

$$\text{ReachabilitySet}(c \in C) = \{t \in C : \text{Classifies}(c, t)\} \quad (1)$$

$$\text{CoverageSet}(c \in C) = \{t \in C : \text{Classifies}(t, c)\} \quad (2)$$

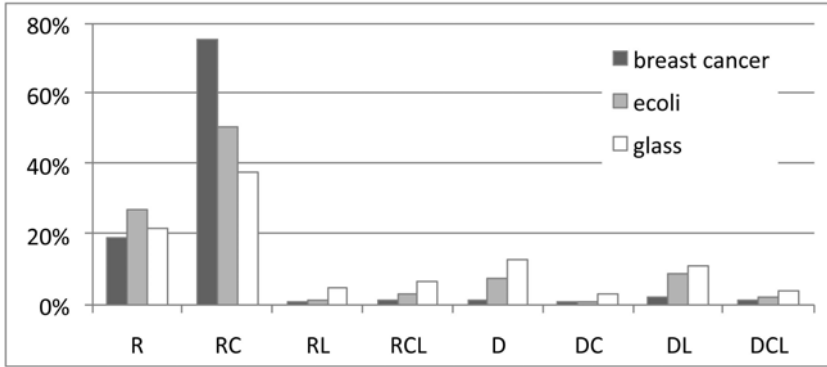
$$\text{LiabilitySet}(c \in C) = \{t \in C : \text{Misclassifies}(t, c)\} \quad (3)$$

In the above equations  $\text{Classifies}(t, c)$  means that case  $c$  contributes to the correct classification of target case  $t$ . This means that target case  $t$  is successfully classified and case  $c$  is returned as a nearest neighbour of case  $t$  and has the same classification as case  $t$ .  $\text{Misclassifies}(t, c)$  means that case  $c$  contributes in some way to the incorrect classification of target case  $t$ . In effect this means that when target case  $t$  is misclassified by the case-base, case  $c$  is returned as a neighbour of  $t$  but has a different classification to case  $t$ . For  $k$ -NN with  $k = 1$ , case  $c$  causes the misclassification but for  $k > 1$  case  $c$  contributes to the misclassification. Case  $t$  is therefore a member of the liability set of case  $c$ .

We propose to further extend the competence properties of a case to include an additional property, the *dissimilarity set*, which complements the reachability set in the same way as the liability set complements the coverage set. The dissimilarity set of a case  $c$  is the set of cases that misclassify case  $c$  and can be represented by Equation 4.

$$\text{DissimilaritySet}(c \in C) = \{t \in C : \text{Misclassifies}(c, t)\} \quad (4)$$

The first point to note about these sets is that one of the reachability set or the dissimilarity set will always be empty. In effect, if we consider that a set exists for a case only if that set is non empty, then the reachability set and the dissimilarity set are mutually exclusive. If a case has a non empty reachability set then it has been classified correctly by the case-base and, as such, will have an empty dissimilarity set and vice versa. However, a case can have one, none or both of the coverage and liability sets. The coverage set of a case  $c$  identifies the potential benefit or usefulness of  $c$  in the case-base, represented by the cases that  $c$  contributes to classifying correctly. On the other hand the liability set of a case  $c$  identifies the damage or harm that  $c$  causes in the case-base represented by the cases that it causes to be misclassified. It is possible for a case to be both useful for some targets and damaging for others.



**Fig. 1.** Composition of datasets with different generalisation accuracies showing the proportion of cases of each profile type. *Breast Cancer* has 95% 10-fold cross validation accuracy, *ecoli* has 81% accuracy while *glass* has 66% accuracy.

### 3.2 Categorising Cases

This leads us to being able to associate an individual case profile with each case in a case-base. We call the profile the *RDCL* profile of a case; it is derived from the competence model of the case-base and includes three characteristics:

- (i) Firstly, it is possible to indicate whether the case is correctly or incorrectly classified by the case-base. This is identified by the case having either a reachability set (R) or a dissimilarity set (D).
- (ii) Secondly, we can consider whether the case is useful, by the existence of a coverage set (C), and
- (iii) Finally whether the case is harmful and causes damage, by the existence of a liability set (L).

Taking all possible combinations of these characteristics, each case in a case-base can have one of eight different case profiles as described below. Fig 1 helps to interpret these by giving the proportion of the different case profiles in different case-bases.

- R** A case which is correctly classified but is not used for classifying any other case in the case-base.
- D** A case which is misclassified but is not used for classifying any other case in the case-base.
- RC** A case which is correctly classified and is useful in that it has contributed to the correct classification of other cases in the case-base. This profile and the R profile are generally the majority case profile types as illustrated in Fig 1.
- RL** A case which is correctly classified but is harmful in the case-base causing damage by contributing to other cases being misclassified.
- DC** A case which is misclassified but is useful in the case-base.

- DL** A case which is misclassified and is harmful in the case-base, (more of these occur in the case-base with poorer generalisation accuracy in Fig 1).
- RCL** A case which is correctly classified and is both useful and harmful in the case-base.
- DCL** A case which is misclassified and is both useful and harmful in the case-base.

## 4 Experimental Analysis

The ability to associate a competence case profile with each case in a case-base offers the opportunity to investigate the structure of case-bases at a case level and the effect of removing different types of cases from a case-base. This section outlines a number of different investigations and evaluations performed using case profile information on a variety of datasets. The datasets used throughout this paper are listed in Table 1 with a description of their characteristics. All datasets are available in the UCI repository [24].

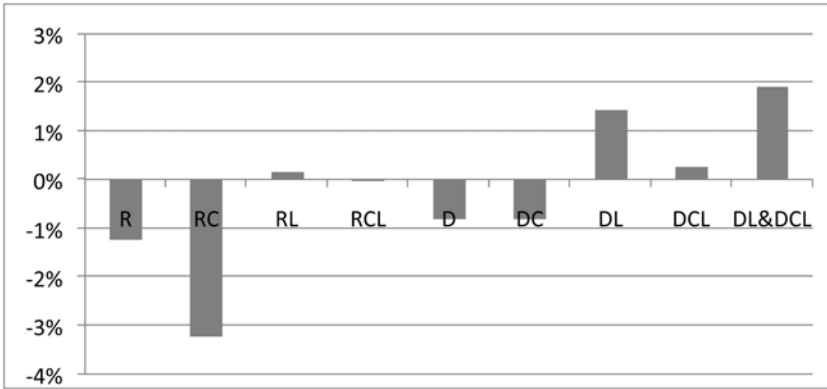
Table 1 also includes the baseline 10-fold cross validation accuracy achieved on each dataset using a  $k$ -NN classifier with a Euclidean distance measure and  $k = 1$ . As the objective is to consider the effect of different case editing strategies,  $k = 1$  was selected as the effect of noise in the data will be more evident with this value since higher values of  $k$  are more noise tolerant.

### 4.1 Removal of Different Types of Cases

Table 2 shows the effect of removing all the cases of each different type of case profile from each dataset. The accuracy was calculated using 10-fold cross validation, using the same folds as the original baseline accuracy given in Table 1. A competence model was build on each training set of nine folds and the training

Table 1. Datasets

	#cases	#classes	#features	class distribution(%)	cv accuracy(%)
breastcancer	683	2	9	65/35	95.5
cmc	1473	3	9	28/16/56	43.3
glass	214	6	9	33/8/35/6/4/14	65.9
musk2	6598	2	166	15/85	73.8
waveform	5000	3	21	33/33/33	77.2
spectf	267	2	44	21/79	71.5
ecoli	336	8	7	42/23/15/10/6/2/1/1	81.0
wine	178	3	13	33/40/27	94.9
ionosphere	351	2	34	68/32	86.0
hill-valley	1212	2	100	50/50	59.4



**Fig. 2.** Accuracy difference, averaged across all datasets, between the unedited dataset and the dataset edited with all cases of the specified case profile removed

set was edited to remove all cases with the specified case profile. The cases in the remaining fold were classified using the edited training set.

Fig 2 shows the difference, averaged across all datasets, of removing cases with the specified profile. This figure illustrates some interesting facts which are discussed below:

**DL/DCL Cases:** The cases that cause the greatest improvement in accuracy by their removal are the DL cases. DL cases are cases that are misclassified by the rest of the case-base and are themselves also causing harm as they are misclassifying other cases. It is to be expected that removing cases such as these would have a beneficial effect on the generalisation accuracy. Considering the individual results in Table 2, the removal of the DL cases is damaging in just one of the datasets where the decrease in accuracy is less than 1%.

DCL cases are somewhat similar to DL cases. The only difference is that these cases, in spite of being misclassified and doing harm, also do some good in that they are used to correctly classify other cases. Looking at the individual dataset results the removal of these cases doesn't typically decrease generalisation accuracy, in all cases but one accuracy remains constant or increases. Overall the effect is beneficial, albeit marginal. This suggests that we are better off without these cases.

Considering the beneficial effect of removing DC and DCL cases separately, Fig 2 and Table 2 also include the results of removing both DC and DCL cases from each dataset. This only has the effect of decreasing, marginally, the accuracy for one dataset, glass. It also results in a higher average increase than removing cases of either profile.

**R Cases:** Cases with an R profile are cases that are classified correctly by the rest of the case-base but are not used in the classification of any other case in that case-base. This suggests that these cases are redundant cases and

**Table 2.** Accuracy values (%) on editing the datasets by removing cases with a specific case profile. Accuracy values which show an increase over the baseline are highlighted in bold. Decreases in generalisation accuracy are in italic.

Dataset	R	RC	RL	RCL	D	DC	DL	DCL	DL&DCL
breastcancer	95.5	<i>94.4</i>	<b>95.9</b>	95.5	<b>95.9</b>	<b>95.6</b>	<b>96.1</b>	95.5	<b>95.9</b>
cmc	<i>42.6</i>	<i>42.8</i>	<i>42.9</i>	<i>42.8</i>	<i>42.8</i>	<i>42.7</i>	<b>46.2</b>	<b>44.1</b>	<b>47.0</b>
glass	65.9	<i>62.2</i>	<b>66.4</b>	<i>64.5</i>	<i>65.4</i>	65.9	<i>65.0</i>	65.9	<i>65.4</i>
musk2	<i>72.4</i>	<i>70.5</i>	<b>74.0</b>	<b>74.5</b>	<b>74.9</b>	<i>73.7</i>	<b>74.9</b>	73.8	<b>74.8</b>
waveform	<i>76.7</i>	<i>74.6</i>	77.2	<i>76.9</i>	77.2	<i>77.0</i>	<b>78.0</b>	<b>77.6</b>	<b>78.5</b>
spectf	<i>70.4</i>	<i>67.8</i>	<b>71.9</b>	<b>72.7</b>	<b>72.7</b>	<i>70.8</i>	<b>73.4</b>	71.5	<b>74.2</b>
ecoli	<i>78.1</i>	<i>77.1</i>	<i>80.4</i>	<b>81.0</b>	<b>82.1</b>	<i>80.1</i>	<b>83.3</b>	81.0	<b>83.0</b>
wine	<i>93.8</i>	<i>93.8</i>	94.9	94.9	<i>94.4</i>	94.9	94.9	94.9	94.9
ionosphere	<b>87.2</b>	<i>81.5</i>	<b>86.3</b>	<b>86.3</b>	<i>85.8</i>	<i>84.6</i>	<b>86.3</b>	<b>86.6</b>	<b>86.9</b>
hill-valley	<i>57.7</i>	<i>59.3</i>	<b>59.7</b>	<b>59.7</b>	<i>53.1</i>	<i>57.8</i>	<b>59.5</b>	<i>59.1</i>	<b>59.7</b>

are not needed. Removing such cases should show no change in generalisation accuracy. Interestingly enough, the removal of these cases causes a decrease in generalisation accuracy on average, with only one of the datasets showing an actual increase in accuracy. This indicates that these cases, although not used to classify the training data, are useful in the classification of unseen data and necessary to maintain good generalisation accuracy. This suggests that R cases may be outlier cases, not well covered by other cases and should not be removed from the case-base.

**D/DC Cases:** Cases with a D profile are cases that are misclassified by the rest of the case-base but are not used in the classification of any other case in that case-base. Corresponding to the reasoning above for the R cases, it might be considered that these cases are redundant cases, as they are not used for classification and could be removed. As these cases are also misclassified, there is an even stronger impetus to remove them from the case-base. However, as Fig 2 shows, overall the removal of these cases has a surprisingly detrimental effect on the generalisation accuracy. In only four of the datasets is the generalisation accuracy increased. This suggests, analogous to the R cases, that D cases may be border cases, situated near the decision boundary which are not well covered by other cases in the case-base.

DC cases are cases that are themselves misclassified by the case-base but that are useful as they contribute to the correct classification of other cases. Similar to the D cases, the removal of these in general decreases the generalisation accuracy of the case-base with a marginal increase of 0.1% shown in only one dataset. This suggests that these cases should not be deleted.

It is also interesting to note that both these types of case, D and DC, are removed by the standard Wilson noise reduction technique used by many case editing algorithms.

**RC Cases:** RC cases are cases that are correctly classified and are used to correctly classify other cases. It is to be expected that the removal of these cases would have a detrimental effect on the generalisation accuracy which we can see is the case. In fact all datasets show a decrease in generalisation accuracy by removing RC cases, with some very significant harmful effects including a drop of over 5% for the ionosphere, spectf and glass datasets.

**RL/RCL Cases:** RL cases are classified correctly but cause harm contributing to the misclassification of other cases. RCL cases are the same but also do good by contributing to the correct classification of other cases. This suggests that these cases are border cases, but considering Fig 2 and Table 2 there is no strong evidence to support the removal of such cases.

It is interesting to note here that both these types of cases, RL & RCL, are removed by another noise reduction algorithm, the BBNR algorithm.

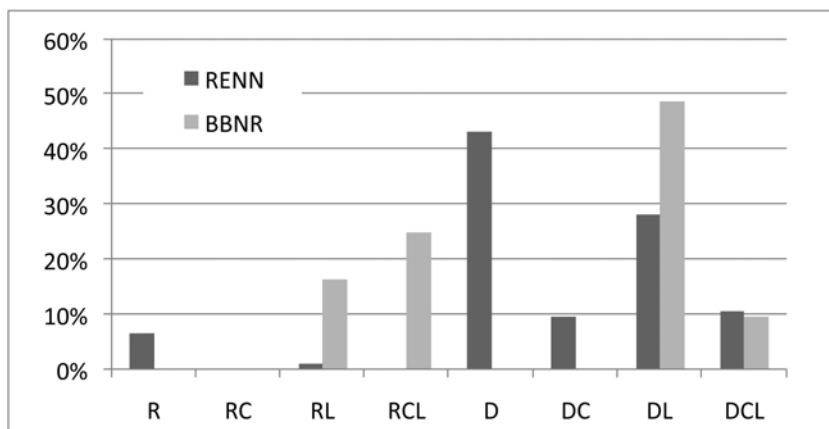
## 4.2 What Existing Noise Reduction Algorithms Do

The original ENN noise reduction technique proposed by Wilson [11] is the algorithm upon which the noise reduction phases of many of the existing case-base editing techniques are based, with a number of them using RENN as a noise reduction stage [3,22,23]. Wilson’s technique removes cases that would be misclassified by the other cases in a training set, assuming that these are incorrectly labelled and are therefore noisy cases. In terms of our profiling, this would be cases that have a dissimilarity set. The later BBNR approach [2] focusses more on the ‘unhelpful’ or harmful cases that *cause* misclassification, i.e. the cases with a liability set. The differences between the principles behind these approaches is evident in Fig 3 which shows for both the RENN and the BBNR algorithms the proportion of deleted cases that were of each case profile type, averaged across all datasets.

RENN removes all cases which are misclassified, which means they contain a D in their profile as they have a dissimilarity set (i.e. it removes 100% of cases with a D, DC, DC or DCL profile). As RENN is an iterative algorithm repeating until no more changes are made to the dataset, it can also remove cases with profiles other than those containing D. This can be seen in Fig 3 where on average 7.3% of the deleted cases have profile R with smaller percentages for RL, RC and RCL cases. Overall, RENN removes on average 10% of the total R cases, 20% of the RL cases, less than 1% of the RC cases and almost 2% of the RCL cases. A high proportion of the cases removed by RENN are, on average, D or R cases (just over 50%); we saw from our analysis of case profiles that removing these cases tends on average to have a bad effect on generalisation accuracy.

BBNR removes all cases which do harm, meaning they have L in their profile, as they contain a liability set. It is obvious from Fig 3 that there is only a small likelihood of overlap in the cases deleted by the two algorithms. If we follow our conclusions from section 4.1 above, BBNR focusses more on the types of cases that are beneficial to generalisation accuracy than RENN (59% of the deleted cases are DL and DCL cases for BBNR whereas 39% are for RENN). In section 4.3 below,





**Fig. 3.** Proportion of deleted cases that are of the specified profile for each noise reduction algorithm averaged across all datasets

we investigate the effect of implementing both algorithms on the selected datasets and will see that RENN has quite inconsistent behaviour which may be due to its focus on case types which are not beneficial to generalisation accuracy.

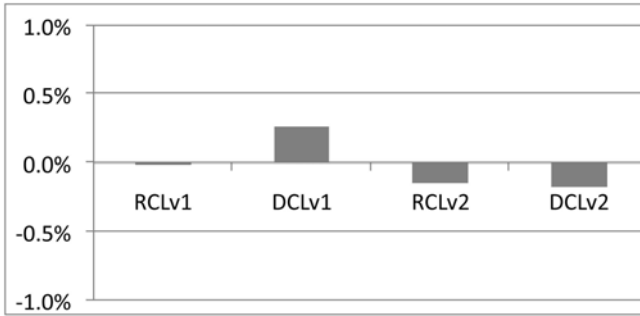
It is also worth noting that neither algorithms removes the RC cases, which our investigations show would have a bad impact on accuracy if removed.

The BBNR algorithm is straightforward in dealing with cases that just do damage, regardless of how they are classified. These cases, DLs and RLs are simply removed. Where a case has a liability set but also a coverage set the BBNR algorithm adopts the principle of ‘not causing even more harm’. BBNR only removes a DCL or RCL case if its removal will not cause even more harm, in other words all cases in its coverage set will still be classified correctly if the DCL or RCL case is removed. This is evident in our experiments, as although 100% of DL and RL cases are removed by BBNR, only on average 55% and 61% of DCL and RCL cases respectively are removed.

We ran an experiment to see the effect of this ‘not causing even more harm’ principle. We only removed a DCL or RCL case if its removal did not result in any of its coverage set being misclassified. The overall results are displayed in Fig 4. It shows that the effect of keeping some of the DCL and RCL cases, those that cause even more harm (labelled v2 in Fig 4) has a worse effect, albeit small, than removing all of them. This raises questions about the benefit of this aspect of the BBNR algorithm.

### 4.3 Comparison of Editing Algorithms

The analysis shown in Fig 2 suggests another noise reduction algorithm, the identification and removal of the DL and DCL cases in a dataset. In this section we compare the performance of existing noise reduction algorithms, RENN and BBNR and this new algorithm. Table 3 presents the results of a 10-fold cross



**Fig. 4.** Percentage accuracy difference, averaged across all datasets, between the unedited datasets and the datasets edited with the specified case profile removed; v1 figures show the result of deleting all cases with the specified profile, v2 figures show the results of keeping those profile cases that cause even more harm by their removal

validation accuracy (on the same folds) across these noise reduction algorithms. The statistical significance of the differences in accuracy of each algorithm against the unedited alternative was calculated using McNemar’s test [25]. Where there were small levels of disagreement (15 cases or less) an exact sign test was used.

The main point to note here is the more consistent performance of removing just DL & DCL cases from the datasets. In just one dataset the generalisation accuracy of the dataset reduces; in all others but one the accuracy actually increases. Compare this with the performance of RENN. For a number of

**Table 3.** Comparison between different noise reduction algorithms. Algorithms which show an increase in accuracy over the unedited case-base are highlighted in bold. Decreases in generalisation accuracy are in italic. Differences significant at the 95% level using McNemar’s test [25] are highlighted with an asterisk.

	Unedited cv accuracy (%)	RENN cv accuracy (%)	BBNR cv accuracy (%)	DL & DCL cv accuracy (%)
breastcancer	95.5	<b>96.6*</b>	<b>96.5*</b>	<b>95.9</b>
cmc	43.3	<b>46.1*</b>	<b>45.1*</b>	<b>47.0*</b>
glass	65.9	<i>63.1</i>	<i>65.4</i>	<i>65.4</i>
musk2	73.8	<b>76.0*</b>	<b>75.1*</b>	<b>74.8*</b>
waveform	77.2	<b>78.9*</b>	<b>78.4*</b>	<b>78.5*</b>
spectf	71.5	<b>75.3</b>	<b>71.9</b>	<b>74.2</b>
ecoli	81.0	<b>85.7*</b>	<b>82.7</b>	<b>83.0*</b>
wine	94.9	<i>93.8</i>	94.9	94.9
ionosphere	86.0	<i>84.9</i>	<b>86.9</b>	<b>86.9</b>
hill-valley	59.4	<i>48.5*</i>	<i>58.3</i>	<b>59.7</b>
<b>Avg Diff</b>		-0.1%	1.0%	1.9%

the datasets the generalisation accuracy is in fact higher for RENN than for DL&DCL, with a highest difference of 2.7% for the ecoli dataset. However the performance of the RENN algorithm is not consistent with considerable decreases in generalisation accuracy for a number of datasets, including a significant 18% drop for hill-valley and more than 3% for glass and ionosphere.

The performance of BBNR seems better, recording only a lower generalisation accuracy over the unedited case-base in two datasets. However, removing DC & DCL cases records equivalent or higher generalisation accuracy than BBNR for all but one of the datasets.

## 5 Conclusions and Future Work

A methodology for categorising cases in a case-base into individual case profile types was presented in this paper. The profile is based on three characteristics that are derived from constructing a competence model of the case-base. These characteristics indicate whether the case has been classified correctly or not, whether the case is helpful to the case-base by its inclusion and/or whether it causes damage in the case-base. Using these profiles we investigated the effect of removing the different types of cases from the case-base. Based on this analysis a simple noise reduction algorithm based on case profiles, was proposed which was more consistent at improving generalisation accuracy on a number of evaluation datasets than the other noise reduction techniques considered.

The ability to categorise cases within a casebase allowed the identification of the types of cases that are removed by existing noise reduction algorithms. Knowing the effect of removing different types of cases, we were able to identify the aspects of the algorithms that contribute to good performance and those that do not.

The work presented in this paper offers opportunities for further work in a number of directions. With a case profiling strategy available, we hope to gain further insights into other case editing algorithms, such as Massie et al. [21]'s TER algorithm, investigating exactly which types of cases are removed. We would like to investigate further into the case profiles, by quantifying the good and harm performed by the cases to allow prioritising cases within a profile for deletion.

## Acknowledgements

The author is grateful to Pádraig Cunningham for discussions about this work. This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/RFP/CMSF718.

## References

1. Smyth, B., Keane, M.: Remembering to forget: A competence preserving case deletion policy for CBR systems. In: Mellish, C. (ed.) Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI, pp. 337–382. Morgan Kaufmann, San Francisco (1995)

2. Delany, S.J., Cunningham, P.: An analysis of case-based editing in a spam filtering system. In: Funk, P., González-Calero, P. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 128–141. Springer, Heidelberg (2004)
3. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery* 6, 153–172 (2002)
4. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. *Machine Learning* 38, 257–286 (2000)
5. Hart, P.E.: The condensed nearest neighbor rule. *IEEE Transactions on Information Theory* 14, 515–516 (1968)
6. Ritter, G.L., Woodruff, H.B., Lowry, S.R., Isenhour, T.L.: An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory* 21, 665–669 (1975)
7. Gates, G.W.: The reduced nearest neighbor rule. *IEEE Transactions on Information Theory* 18, 431–433 (1972)
8. Chou, C.H., Kuo, B.H., Chang, F.: The generalized condensed nearest neighbor rule as a data reduction method. In: ICPR 2006: Proceedings of the 18th International Conference on Pattern Recognition, Washington, DC, USA, pp. 556–559. IEEE Computer Society, Los Alamitos (2006)
9. Angiulli, F.: Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering* 19, 1450–1464 (2007)
10. Hao, X., Zhang, C., Xu, H., Tao, X., Wang, S., Hu, Y.: An improved condensing algorithm. In: Seventh IEEE/ACIS International Conference on Computer and Information Science, ICIS 2008, pp. 316–321 (2008)
11. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics* 2, 408–421 (1972)
12. Tomek, I.: An experiment with the nearest neighbor rule. *IEEE Transactions on Information Theory* 6, 448–452 (1976)
13. Sánchez, J.S., Barandela, R., Marqués, A.I., Alejo, R., Badenas, J.: Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters* 24, 1015–1022 (2003)
14. Jiang, Y., Zhou, Z.: Editing training data for knn classifiers with neural network ensemble. In: Yin, F.-L., Wang, J., Guo, C. (eds.) ISSN 2004. LNCS, vol. 3173, pp. 356–361. Springer, Heidelberg (2004)
15. Koplowitz, J., Brown, T.A.: On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition* 13, 251–255 (1981)
16. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. *Machine Learning* 6, 37–66 (1991)
17. Brodley, C.: Addressing the selective superiority problem: Automatic algorithm/mode class selection. In: Proceedings of the 10th International Conference on Machine Learning (ICML 1993), pp. 17–24. Morgan Kaufmann Publishers Inc., San Francisco (1993)
18. Cameron-Jones, R.M.: Minimum description length instance-based learning. In: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, pp. 368–373. Morgan Kaufmann Publishers Inc., San Francisco (1992)
19. Zhang, J.: Selecting typical instances in instance-based learning. In: Proceedings of the 9th International Conference on Machine Learning (ICML 1992), pp. 470–479. Morgan Kaufmann Publishers Inc., San Francisco (1992)
20. Pan, R., Yang, Q., Pan, S.J.: Mining competent case bases for case-based reasoning. *Artificial Intelligence* 171, 1039–1068 (2007)

21. Massie, S., Craw, S., Wiratunga, N.: When similar problems don't have similar solutions. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 92–106. Springer, Heidelberg (2007)
22. McKenna, E., Smyth, B.: Competence-guided editing methods for lazy learning. In: Horn, W. (ed.) ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, pp. 60–64. IOS Press, Amsterdam (2000)
23. Wilson, D., Martinez, T.: Instance pruning techniques. In: ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 403–411. Morgan Kaufmann Publishers Inc., San Francisco (1997)
24. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
25. Dietterich, D.T.: Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computing* 10, 1895–1923 (1998)

# An Active Approach to Automatic Case Generation

Michael W. Floyd and Babak Esfandiari

Department of Systems and Computer Engineering  
Carleton University  
1125 Colonel By Drive  
Ottawa, Ontario

**Abstract.** When learning by observing an expert, cases can be automatically generated in an inexpensive manner. However, since this is a passive method of learning the observer has no control over which problems are solved and this can result in case bases that do not contain a representative distribution of the problem space. In order to overcome this we present a method to incorporate active learning with *learning by observation*. Problems that are not covered by the current case base are automatically detected, during runtime or by examining secondary case bases, and presented to an expert to be solved. However, we show that these problems can not be presented to the expert individually but need to be part of a sequence of problems. Creating this sequence of cases is non-trivial, and an approach to creating such sequences is described. Experimental results, in the domain of simulated soccer, show our approach to be useful not only for increasing the problem coverage of the case base but also in creating cases with rare solutions.

## 1 Introduction

In case-based reasoning (CBR), the solutions to novel problems are determined using the solutions of previously encountered problems. These previously encountered cases are crucial to the problem solving ability of CBR systems, so it is important that cases be of a high quality and representative of the entire problem space. The initial set of cases used by a CBR system is typically provided by an expert, either manually authored or transferred in another manner. However, having an expert manually author cases can be an expensive task and requires the expert to be able to encode their knowledge in case form. Another approach to transferring knowledge from an expert into cases is to have a system that *learns from observation*.

In existing CBR systems that learn from observing an expert [1,2,3] cases are generated by remembering how the expert behaves (its outputs) in response to sensory stimuli (its inputs). A limitation of such a passive learning approach is that the learnt cases are directly related to the observed behaviour of the expert. If the expert does not encounter specific problems while being observed

then there will be no cases created related to those problems. Even if the expert is observed for an extended period of time, or on multiple occasions, there is no guarantee that a representative sample of the entire problem space will be encountered. If there is no way to directly interact with the expert, like asking for a specific problem to be solved, then there can exist areas of the problem space that are not represented in the case base.

We examine a hybrid approach that incorporates active learning in order to explore areas of the problem space that are not represented in a case base that was created using passive learning. When a problem is identified that is not sufficiently similar to any existing case in the case base, that problem is artificially<sup>1</sup> presented to the expert. The resulting actions of the expert can then be assumed to be the solution to that problem and a new case can be added to the case base.

A flaw with this simple approach to active case learning is that it assumes that the solution provided by the expert is only dependant on the currently encountered problem. If a temporal link exists between problems [4], such that a set of problems can be ordered based on the time they are encountered, then the solution to a problem can instead be a function of the current problem as well as several previous problems. Such a situation can occur when the expert maintains an internal model of the world. If problems are not presented to the expert in the proper order then the world model may not be properly built, resulting in the expert reacting differently than if the problems had been presented in the correct order.

In order to overcome this, we look to estimate a set of problems that were likely to have occurred before the problem of interest. Initially, the case from the case base that is most similar to the problem is found. A series of problems that connect the similar case to the problem of interest are then created. This is done by performing a series of alterations to the case such that each new intermediate problem is slightly more similar to the problem of interest, compared to the previously created intermediate problem. The entire series of problems can then be sequentially presented to the expert. As an added benefit, the solutions to these intermediate problems will also be determined and can therefore be added to the case base.

The remainder of this paper will present an approach for actively acquiring cases when cases are learnt by observing a teacher. In Section 2, work related to automatic case generation and learning from observation is presented. Section 3 deals with capturing an expert's behaviour in cases. Approaches for identifying problems that may be of interest and actively acquiring their solutions is discussed in Section 4. Next, Section 5 presents a method to generate a sequence of intermediate cases that link two cases. Section 6 details experimental results and Section 7 provides conclusions and directions for future work.

---

<sup>1</sup> In a simulated environment this would involve altering the input messages sent to the expert. In a physical environment it would require altering the sensory inputs of the expert, like with a virtual reality system.

## 2 Related Work

In our previous work, we demonstrated how a soccer playing agent can learn, using CBR, by observing the behaviour of another agent [1]. While we examined how the case base can be preprocessed to select representative cases [5], the cases were created in a passive manner. Similarly, Romdhane and Lamontagne [2] have used case-based reasoning to teach an agent how to play the game of Tetris by observing an experienced player. In a real-time strategy domain, Ontañón et al. [3] build cases for use in case-based planning by watching a human. Flinter and Keane [6] automatically extract cases from logs of grandmaster chess matches, so their CBR system attempts to play chess like the grandmaster would. Much like our work, these approaches all collect cases in a passive manner so the CBR systems have no control over the problem space covered by the resulting case base.

Learning from observation has also been explored using learning methods other than case-based reasoning. Atkeson and Schaal [7] present a method for teaching a robotic arm to rotate and balance a pendulum by watching a human perform the task. Similarly, Coates et al. [8] teach a robotic helicopter aerobatic manoeuvres by having a human control the robot during a series of demonstrations. A common experimental result in these works is that while the robots are able to perform parts of the learnt tasks well, there exist parts that are difficult to perform. If the learners were able to actively produce more data, in these problem areas, they might be able to improve their ability to perform those tasks. Grollman and Jenkins [9] attempt to overcome this by allowing a soccer playing robot to be simultaneously controlled by a human and an autonomous system. If the human is actively controlling the robot the autonomous system learns by comparing what it would have done to what the human actually did. The autonomous system controls the robot otherwise. While this allows for active learning, the autonomous system is not able to automatically detect areas that require further learning and requires an expert to initiate the learning process.

In Yang et al. [10], aviation maintenance cases are generated in an automated manner from a pair of data sources. Text reports, from technicians, as well as computer generated fault messages are mined for data and combined to create cases. Their method places a significant importance on automatically extracting information from text, as does Asiimwe et al. [11] where cases are extracted from reports about home upgrades that help people deal with disabilities. Automatic Case Elicitation (ACE), which has been applied to checkers [12] and chess [13], uses reinforcement learning to rate automatically created cases. Solutions, in the form of which game piece should be moved, are randomly generated and applied to the current game board and a case is created. Any cases created during a winning game gain positive reinforcement, whereas losing games have their cases reinforcement values decreased. This approach allows a measurement of the usefulness of generated cases but requires a way to determine if a case resulted in a positive outcome, which may not always be easily determined.

Active learning has been used in case-based reasoning using measures of complexity [14] and coverage [15]. These approaches are successful in identifying



areas of the case base that are poorly covered but only present individual problems to the expert to solve which may not be applicable if the expert reasons using information from a series of past problems. The method we use to create a series of connecting cases is similar to the idea of adaptation paths [16]. Adaptation paths are used to create a series of slightly different problems in order to transform the initial problem into a problem with a known solution. These adaptation paths are not presented to the expert to be solved, but are instead intermediate steps used to trace out the logic used during adaptation.

### 3 Modelling an Expert's Behaviour

When using a case-based reasoning system to learn from observation, the goal is to determine how the expert behaves in response to the state of the environment. A case,  $C$ , can then be defined as a tuple containing the sensory stimulus,  $S$ , received by the expert and the corresponding actions,  $A$ , performed by the expert.

$$C = (S, A)$$

During a period of observation a series of  $N$  cases will be learnt from the expert, with a temporal relationship existing between these cases. Since each case represents the expert's stimulus at a moment in time, and subsequently performed actions, the  $i$ th case will have been observed before the  $(i + 1)$ th case.

Our existing representation of a case, however, assumes the expert behaves in a purely reactive manner. The actions of the expert are only considered to be a function of the current stimulus,  $A_i = f(S_i)$ , so no information about the preceding stimuli is included. If the expert does not simply react to the current stimulus but maintains an internal model of the world then the cases will not contain all of the information that the expert reasons with. Thus, the actions of the expert are actually a function of the current stimulus as well as the  $k$  previously encountered stimuli ( $A_i = f(S_i, S_{i-1}, \dots, S_{i-k})$ ). If these previously encountered stimuli are changed, but the current stimulus remains the same, then the actions performed by the expert may change.

This introduces the need to have the cases ordered and for a case to be aware of the cases that precede it. The definition of a case can then be extended to include a timestamp,  $T$ , that is used to provide a temporal ordering to the cases. This allows a case to identify and use information from preceding cases.

$$C = (S, A, T)$$

An alternate approach, which would remove the need for a timestamp, would be to have each case include the  $k$  preceding cases. While such an approach would encapsulate all of the information needed for reasoning in a single case, it requires knowing how many preceding stimuli are required. If the observer has no information about the expert being watched it will not know how many previous stimuli should be included in each case. For example, a purely reactive

expert would not require any previous stimuli whereas experts who maintain world models would. Using a timestamp allows for a more dynamic approach, and reduces duplication of information, since it is possible to go back (or forward) any number of cases.

## 4 Improving Passive Learning with Active Case Generation

Learning by observation, by its nature, is a passive learning method. The observer watches an expert and attempts to learn the behaviour the expert demonstrates. Interaction occurs between the expert and the environment as the environment produces stimuli that are sensed by the expert and the expert performs actions that influence the environment. As shown in Figure 1, the observer can then view and learn from these interactions. There is no direct interaction between the observer and expert and the expert may not even be aware it is being watched.

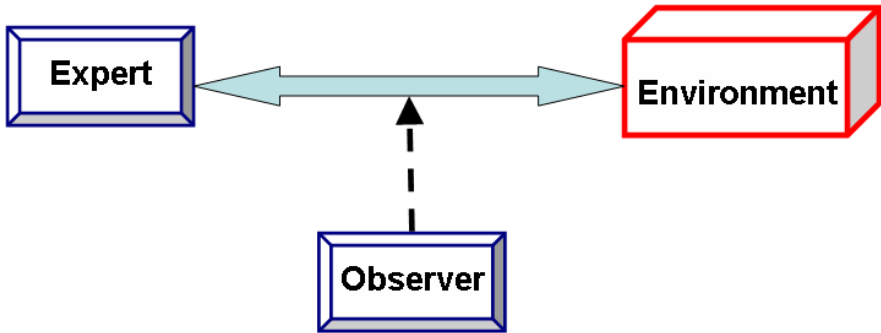


Fig. 1. Passive learning by observing an expert

As was mentioned in the previous section, the observer will create a series of cases while watching the expert. These cases will capture the interactions between the expert and the environment over a period of time. Thus, only the expert and the environment have any control over the cases that are produced. The environment controls which stimuli are contained in the cases and the expert controls the actions. In such a passive approach, even if the observer watches for an extended period of time there is no guarantee that every possible action will be performed or that a representative sample of the possible stimuli will have been sensed.

Ideally, we want the observer to be able to examine the cases it has learnt and identify areas of the problem space that it should explore further. When interesting problems that are not represented in the case base are identified, active learning can be used to complement the passive learning process. During

active learning, the observer can present problems to the expert to solve, thereby gaining a level of control over the contents of the cases. It should be noted that although active learning gives more control over the problems that are solved it is more invasive than passive learning and, as we will see in the next section, requires more computations. Active learning, therefore, will be used as a secondary learning method with the majority of the learning being done using passive learning.

We present two methods that can be used to identify potential problems to be solved using active learning. These methods are not mutually exclusive and can be used in combination or separately.

- **Runtime Identification:** After the observer has learnt a number of cases, it can then use those cases to attempt to imitate the behaviour of the expert. During runtime, the observer will receive a stimulus from its own environment and search its case base for cases with similar stimuli. The actions from these cases are then used to determine an action for the observer to perform. If no cases in the case base are similar enough to the input stimulus, then the observer may not select the correct action to perform. For this approach, during each case base search if no case,  $C_i$ , has a similarity to the input stimulus,  $I$ , above a threshold,  $T$ , then the input stimulus is logged so it can be solved using active learning ( $\forall C_i, sim(I, C_i) < T$ ). This threshold value will influence the number of stimuli that are used for active learning, with higher threshold values resulting in more stimuli being logged.
- **Secondary Case Base:** When learning from observation, different case bases can be created depending on the expert being observed. For each type of expert that is observed a separate case base is created that represents the behaviour of that expert. Two experts may perform the same task, like playing soccer, but may do so in different ways and with different levels of skill. The two experts may react differently when presented with the same stimuli, so it may not be appropriate to have cases from two different experts in a single case base<sup>2</sup>. Even if these case bases can not be combined directly, it is still possible to extract information from other related case bases. Given two case bases, a primary and secondary, cases from the secondary case base can be compared to those in the primary case base. Similarly to the runtime approach, any cases that have no similar cases in the primary case base can be logged for active learning. Secondly, if the secondary case base has cases with actions that are rare or non-existent in the primary case base those cases can be logged as well. While there is no guarantee that the problems in these cases will result in the expert performing those rare actions, after active learning, it does help guide the search for cases with rare solutions.

A third method that could be applied would be to randomly create problems. This approach, however, is limited in that there is no guarantee of the validity of these randomly created problems. In the previous two approaches, all of the

---

<sup>2</sup> For example, if one expert is a defender and the other is an forward on a soccer team. The observer may only want to behave in an defensive manner.

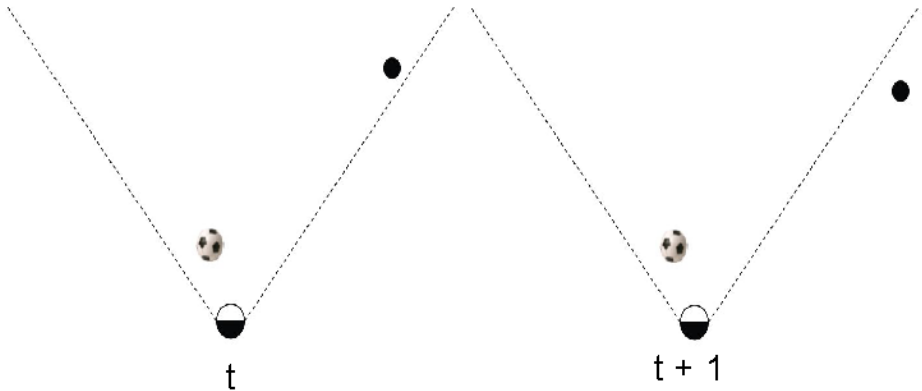
problems have been encountered while observing an expert so these problems are known to be valid. There may be underlying constraints on problems, such as the acceptable values of stimuli, that need to be considered when creating problems. If these constraints were unknown, it would be possible to create problems that are impossible to actually encounter. For example, when observing a soccer playing expert there is a limit to the number of opponents that the expert could ever see in the environment due to the rules of soccer. If this limit was unknown to the observer, a randomly created problem could be created that contained more opponents than are allowed.

## 5 Determining a Connecting Sequence

When a problem is identified for active learning using the techniques described in the previous section, it must be presented to the expert to be solved. The most direct approach would be to present each problem to the expert individually. However, as was described in Section 3, the expert might maintain a world model based on previously encountered problems. Only presenting the expert with a single problem may result in different behaviour than if the expert was given that problem as part of a sequence.

In order to ensure that the expert is able to solve the problem in the proper context, by building a world model before encountering the problem, it becomes necessary to determine a series of problems to present to the expert before the problem of interest. However, for a given problem, a series of preceding problems may not be known. We look to determine these unknown preceding problems using the following method:

1. For a problem,  $P_1$ , find the most similar case,  $C$ , in the case base.
2. Extract the problem,  $P_2$ , from  $C$ .



**Fig. 2.** Changing number of objects visible in the expert's field of vision

3. Determine a series of connecting problems,  $L$ , such that the stimuli change by no more than  $\alpha$  between the  $i$ th and  $(i + 1)$ th problems in  $L$ . The  $\alpha$  value represents the percentage of change between the stimuli. For example, the position of an object would change by at most  $\alpha$  percent between problems.

The goal of this is to minimize the number of connecting problems that must be created, by starting with a problem that is similar to the final problem, and to gradually change the stimuli. Stimuli are changed gradually so there are no sudden large changes in what the expert senses. For example, if the visual stimulus received by the expert changed drastically it might appear like visible objects are suddenly changing locations.

We further decompose the stimulus received by the expert,  $S$ , into a collection of individual stimuli. For example, these stimuli might be visible objects, sounds, touch, or other sensory inputs. We define each stimulus,  $S_i$ , to be composed of  $k_i$  sub-stimuli ( $S_i = \{s_{i1}, \dots, s_{ik_i}\}$ ). Each stimulus potentially having a different number of sub-stimuli is due to the fact that the expert may not have a complete view of the environment. The expert will only sense a subset of the possible stimuli, with the remaining stimuli being unknown to the expert. For example, in Figure 2 we can see that objects can move out of (or into) the experts field of vision, thereby changing the number of stimuli the expert can sense.

In Algorithm 1, we describe how a series of connecting problems can be created that link together start and end problems. Each stimulus,  $s_s$ , from the start problem is matched<sup>3</sup> with a stimulus,  $s_e$ , from the end problem. The start stimulus is then modified, by a maximum of  $\alpha$ , to be more like the end stimulus. This modified stimulus,  $s_c$ , is then added to the connecting problem that is currently being constructed,  $P_c$ . If  $P_c$  is equal to the end problem, the algorithm terminates. Otherwise,  $P_c$  is added to the series of connecting problems and is then used recursively as the next start problem.

Unlike when problems are randomly created, the connecting problems will be guaranteed to have a valid number of stimuli of each type. This is because the number of stimuli will be bound between the number in the start problem and the number in the end problem. For example, if  $P_s$  contained 5 stimuli and  $P_e$  contained 2 stimuli then all connection problems would contain between 2 and 5 stimuli. However, no testing is performed to ensure the validity of the relations between the stimuli. For example, all flags must be a fixed distance apart from each other but the algorithm never tests to ensure this is true in connecting cases. Future work will involve identifying these rules and forcing connecting cases to follow them.

## 6 Experimental Results

Due to the possibility that the start and end problems can have different numbers of stimuli, two situations can arise. First, if there are more start stimuli than end stimuli then not all start stimuli will have a match. The *modify* function will

---

<sup>3</sup> A detailed description of how stimuli can be matched is described in [1].

**Algorithm:**  $L = \text{connectors}(P_s, P_e, \alpha)$   
**Data:** start problem  $P_s$ , end problem  $P_e$ , maximum change  $\alpha$   
**Result:** the series of connecting problems  $L$   
 $L = \{\emptyset\};$   
 $P_c = \{\emptyset\};$   
 $M = P_e;$   
**foreach**  $s_s \in P_s$  **do**  
     $s_e = \text{match}(s_s, M);$   
     $M -= s_e;$   
     $s_c = \text{modify}(s_s, s_e, \alpha);$   
     $P_c += s_c;$   
**end**  
**foreach**  $s_e \in M$  **do**  
     $s_c = \text{modify}(\emptyset, s_e, \alpha);$   
     $P_c += s_c;$   
**end**  
**if**  $P_c == P_e$  **then**  
    return  $\emptyset;$   
**else**  
     $L = P_c + \text{connectors}(P_c, P_e, \alpha);$   
    return  $L;$   
**end**

**Algorithm 1.** Determine a series of connecting problems

attempt to remove those stimuli. An example would be to move the position of an object outside the field of vision so it can no longer be seen. Second, when there are more end stimuli than start stimuli it means stimuli needed to be added. This could involve introducing an object at the boundary of the field of vision.

The experiments we perform will attempt to answer the following questions:

- Are there certain experts that require problems to be presented in a specific sequence or can they be presented in a random order?
- Does estimating a series of preceding problems, and presenting that series along with the problem of interest to the expert, help in determining the correct solution?
- Can secondary case bases be mined in order to identify problems that may result in rare solutions?

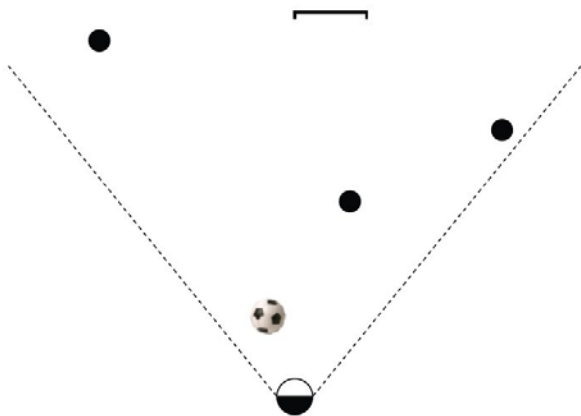
## 6.1 Experimental Setup

The domain we use is simulated RoboCup soccer [17]. In the RoboCup Simulation League, the environment contains objects that belong to a fixed number of *object types*. Although each individual object on the field is unique, an agent is often unable to distinguish between objects of the same type due to noise. For example, the agent would be able to see a teammate but might not be able to tell what specific teammate it is. Additionally, the agent may not care which

specific object it is but only what type of object it is. For these reasons, objects of the same type are treated as interchangeable. In the RoboCup Simulation League we define the following object types:

$$\textit{Type} = \{ \textit{Ball}, \textit{Goalnet}, \textit{Flag}, \textit{Line}, \textit{Teammate}, \textit{Opponent}, \textit{Unknownplayer} \}$$

The stimuli contained in a case are then a collection of objects that are within the expert's field of vision (as shown in Figure 3), and their location relative to the expert. The expert can then perform an action: kick, dash, or turn.



**Fig. 3.** Field of vision of a soccer playing agent

The expert that will be observed is a player from the CMUnited<sup>4</sup> soccer team [18]. CMUnited are the former champions of the RoboCup Simulation League and use a layered learning architecture and a number of strategies including formation strategies. CMUnited players can have multiple states of behaviour and maintain internal models of the world, making them a good candidate to experiment on.

Data was generated by watching the CMUnited team playing against a very simple opposing team<sup>5</sup>, with each team composed of 11 players. Cases were generated by observing complete soccer games, with a total of 25 complete games being observed.

In the RoboCup Simulation League, the players connect to a server that maintains the model of the environment and acts as a referee for the game. The server sends the agent messages that contain information on the objects the agent can

<sup>4</sup> The standard CMUnited source code was modified slightly. The default version of the code would stop functioning if unexpected inputs were given, for example if stimuli were given in a random order.

<sup>5</sup> The team used was Krislet [19] who simply chase the ball around the field and kick it toward the opponents goal.

currently see, and the agent can then send messages to the server about the action it wishes to perform. For the active learning process, we create a fake server for the agents to connect to. The fake server can then send messages to the agent related to the problems that are to be solved, and the resulting messages from the agent can then be logged.

Since each problem used in these experiments will be extracted from a pre-existing case, there will also be a known solution to those problems in the case. The known solution will be compared to the solution generated with active learning and the f-measure will be used to measure the performance. We define the f-measure,  $F$ , for a single action type,  $i$ , as:

$$F_i = \frac{2 \times \textit{precision}_i \times \textit{recall}_i}{\textit{precision}_i + \textit{recall}_i} \quad (1)$$

with

$$\textit{precision}_i = \frac{c_i}{t_i} \quad (2)$$

and

$$\textit{recall}_i = \frac{c_i}{n_i} \quad (3)$$

In the above equations,  $c_i$  is the number of times the known action and generated action matched,  $t_i$  is the total number of times the action was generated and  $n_i$  is the number of times the action should have been generated. The f-measure takes into account how accurately an action is selected (the recall) as well as if when the action is selected it is selected correctly (the precision). The global f-measure, combining the f-measures for all  $A$  actions, is:

$$F_{\textit{global}} = \frac{1}{A} \sum_{i=1}^A F_i \quad (4)$$

## 6.2 Importance of Problem Order

Thus far, the assumption has been made that some experts will only solve problems correctly when given a sequence of problems, not just an individual problem. In order to validate this we modify the ordering of problems presented to the expert during active learning and examine the affects. Two variations of the ordering were examined: a random ordering and the original ordering. The case bases, for each of the 25 complete games, were presented to the expert using each of the orderings with the results presented in Table II.

We can see from the results that there is a large performance difference between using the original ordering and using a random ordering. Using the original ordering, which maintains the temporal ordering of problems, performs significantly better. This verifies our assumption that the CMUnited team relies on past stimuli to maintain an internal world model. In fact, the CMUnited agent would often output warning messages when the random ordering was used since the stimuli it was receiving was changing so drastically. One item of note is that



even when the original ordering was used there were still situations where the expert responded with a different solution than the known solution. A likely reason for this is that the expert has some degree of randomness in its action selection process or relies on other stimuli which are not encoded in the cases.

**Table 1.** Comparison of results

	f-measure	Precision			Recall		
		dash	turn	kick	dash	turn	kick
<b>Original</b>	0.82 (+/- 0.04)	0.85	0.77	0.81	0.89	0.75	0.85
<b>Random</b>	0.54 (+/- 0.11)	0.62	0.44	0.56	0.56	0.42	0.65
<b>With Connecting</b>	0.80 (+/- 0.03)	0.86	0.77	0.75	0.78	0.79	0.85
<b>Without Connecting</b>	0.68 (+/- 0.06)	0.77	0.72	0.58	0.68	0.68	0.67

### 6.3 Applying Active Learning

These experiments aim to demonstrate the benefit of estimating a series of preceding problems and presenting those problems to the expert before the problem of interest. A case is selected at random from among the CMUnited cases, and a sequence of 20 cases is extracted from the case base such that the randomly selected case is at the end of the sequence<sup>6</sup>. A randomly sized sequence of preceding cases, between 1 and 10 cases, were then removed from the sequence. This left the sequence containing the first 9 to 18 cases from the original sequence as well as the last case. The last case and the case that now preceded it were then used to estimate a sequence of problems that connect them. The problem portions of these two cases were used as input to Algorithm 1, with a value of  $\alpha = 5\%$ , in order to generate connecting problems.

Active learning was then used on the sequence, both with the connecting problems included in the sequence and without. Each run of the experiments extracted 100 sequences, and the runs were performed 25 times. The results can be found in Table 1.

With these results, we can see a distinct improvement in performance when connecting problems are created and included in the sequences used in active learning. This is likely due to the fact that the connecting problems help to gradually change the stimuli presented to the expert, rather than cause a large instantaneous change. When connecting problems are used there is only a slight decrease in performance (and not a statistically significant difference using a t-test with  $p=0.05$ ) compared to presenting problems to the expert in the original order they were observed. This tells us that the connecting problems are accurate estimates of the actual problems that preceded a case. We can also see that not using connecting problems performed poorly, but not as poorly as presenting problems in a random order. This is because part of the sequence is preserved, but a gap exists in the sequence that can cause stimuli to change drastically. When the results were examined in more detail, it was found that the method

<sup>6</sup> If there are not that many cases before it, that case is not used.

that did not use connecting sequences performed better when fewer cases were removed from the original sequence due to the smaller gap that was created.

As a second set of experiments, we look to examine other case bases to identify problems with specific solutions. In previous work in learning by observing a RoboCup player [15], the solution space was found to be highly imbalanced (approximately 67.8% dash, 32.1% turn and 0.1% kick). Using passive learning, there would be significantly more *dash* actions compared to *kick* actions. This solution imbalance resulted in difficulty correctly selecting these rare actions. Thus, being able to generate more cases that have these rare solutions might be useful for increasing the performance of the CBR system. Such a solution imbalance could also occur depending on the circumstances under which the expert is observed. For example, if a soccer playing agent is observed playing a game against a far superior opponent it may never get the opportunity to perform certain behaviours like kicking the ball.

Case bases that were created by observing another soccer agent, called Krislet [19], are examined and all cases that have the *kick* action are extracted. While we cannot use these cases directly, since Krislet plays soccer differently than CMUnited and may react differently to stimuli, the problems in these cases can provide a good starting point for active learning. The method for determining connecting case, described in Section 5, was used on the problems in these cases (using  $\alpha = 5\%$ ) and then the resulting sequences of problems are presented to the CMUnited agent. In total, cases from Krislet playing 25 full games of soccer were examined and 883 cases were found that had the *kick* action. Of those, active learning found CMUnited produced the *kick* action in 634 of the problems. Comparatively, selecting 883 cases at random only resulted in CMUnited performing the *kick* action 67 times. We can see that using such a targeted approach helps guide the search for problems with rare solutions.

Identifying these rare actions does not guarantee an improvement in the performance but is likely to improve performance in some situations. In an extreme situation where there were no *kick* actions in the case base, the f-measure for the kick action would be 0 since no test cases would ever be properly classified as *kicks*. By actively obtaining cases with a *kick* action, it would then be possible to properly classify some of the *kick* test cases and thereby increase the f-measure results. This can be thought of as a sampling method, since it helps increase the number of cases for a specific class of action.

## 7 Conclusions and Future Work

Our work has attempted to address the limited control over the problems in a case base when cases are obtained through passive learning, specifically when learning is done by observation. We present an approach that incorporates active learning by identifying problems that are not represented in the case base, either during runtime or by examining other case bases, and presenting those problems to an expert to solve. This approach aims to make the active learning as nonintrusive as possible by making the expert think the problems, made up

of sensory stimuli, are coming from the environment and not from an outside source.

Our results show that problems can not be presented to an expert individually. Instead problems must be provided in the proper context, by giving the expert a sequence of problems, so that the expert can properly build a world model before attempting to solve the problem of interest. This is a result of the inherent temporal link between cases that are learnt from observation. The approach described, where two problems are linked with a series of connecting problems, was found to produce solutions which are highly similar to the expected solutions. Additionally, we show how mining a related case base can be used to identify problems that have solutions which are rare. This technique can be used to help balance the distribution of the solution space by boosting the number of occurrences of rare solutions. Our results show that even when a set of cases that represent a complete soccer game are presented to the expert, in the correct order, that the solution accuracy is not perfect. This could be due to stimuli that are not included in a case, like inter-agent communication, or an amount of randomness in the action selection process.

While this work has shown the benefit of active learning using a series of connecting problems in a simulated soccer domain, the results could be applicable in a variety of domains. Providing the proper context for a problem, in the form of a series of preceding problems, would be applicable when learning from any expert that uses previously solved problems to maintain a world model. Although the limited scope of our experiments do not allow us to draw broader conclusions about the performance benefit of our techniques, our results are promising and future work will examine other domains and non-classification tasks. Another area of interest is examining if there are certain problems that can produce multiple solutions and if the cause of this can be identified. If there exist highly similar problems that have different solutions this might indicate that other information, like previous problems, are being used during reasoning. Also, future work will look at if examining the influence of problem ordering can be used to measure the complexity of an expert's reasoning process in order to determine how long a sequence needs to be given to the expert.

Full results, data sets, sourcecode and videos related to this work are available online<sup>7</sup>.

## References

1. Floyd, M.W., Esfandiari, B., Lam, K.: A case-based reasoning approach to imitating RoboCup players. In: Twenty-First International Florida Artificial Intelligence Research Society Conference, pp. 251–256 (2008)
2. Romdhane, H., Lamontagne, L.: Reinforcement of local pattern cases for playing Tetris. In: Twenty-First International Florida Artificial Intelligence Research Society Conference, pp. 263–268 (2008)

---

<sup>7</sup> <http://rcscene.sf.net>

3. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 164–178. Springer, Heidelberg (2007)
4. Jære, M.D., Aamodt, A., Skalle, P.: Representing temporal knowledge for case-based prediction. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416, pp. 174–188. Springer, Heidelberg (2002)
5. Floyd, M.W., Davoust, A., Esfandiari, B.: Considerations for real-time spatially-aware case-based reasoning: A case study in robotic soccer imitation. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 195–209. Springer, Heidelberg (2008)
6. Flinter, S., Keane, M.T.: On the automatic generation of cases libraries by chunking chess games. In: 1st International Conference on Case-Based Reasoning, pp. 421–430 (1995)
7. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: 14th International Conference on Machine Learning, pp. 12–20 (1997)
8. Coates, A., Abbeel, P., Ng, A.Y.: Learning for control from multiple demonstrations. In: 25th International Conference on Machine Learning, pp. 144–151 (2008)
9. Grollman, D.H., Jenkins, O.C.: Sparse incremental learning for interactive robot control policy estimation. In: IEEE International Conference on Robotics and Automation, pp. 3315–3320 (2008)
10. Yang, C., Farley, B., Orchard, R.: Automated case creation and management for diagnostic CBR systems. *Applied Intelligence* 28(1), 17–28 (2008)
11. Asimwe, S., Craw, S., Taylor, B., Wiratunga, N.: Case authoring: From textual reports to knowledge-rich cases. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 179–193. Springer, Heidelberg (2007)
12. Powell, J.H., Hauff, B.M., Hastings, J.D.: Evaluating the effectiveness of exploration and accumulated experience in automatic case elicitation. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS, vol. 3620, pp. 397–407. Springer, Heidelberg (2005)
13. Powell, J.H., Hastings, J.D.: An empirical evaluation of automated knowledge discovery in a complex domain. In: Workshop on Heuristic Search, Memory Based Heuristics and their Applications: Twenty-First National Conference on Artificial Intelligence (2006)
14. Massie, S., Craw, S., Wiratunga, N.: Complexity-guided case discovery for case based reasoning. In: The Twentieth National Conference on Artificial Intelligence, pp. 216–221 (2005)
15. McSherry, D.: Automating case selection in the construction of a case library. *Knowledge-Based Systems* 13(2-3), 133–140 (2000)
16. Lieber, J., d’Aquin, M., Badra, F., Napoli, A.: Modeling adaptation of breast cancer treatment decision protocols in the kasimir project. *Applied Intelligence* 28(3), 261–274 (2008)
17. RoboCup: Robocup official site (2009), <http://www.robocup.org>
18. Stone, P., Riley, P., Veloso, M.M.: The CMUnited-99 champion simulator team. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup 1999. LNCS, vol. 1856, pp. 35–48. Springer, Heidelberg (2000)
19. Langner, K.: The Krislet Java Client (1999), <http://www.ida.liu.se/~frehe/RoboCup/Libs>

# Four Heads Are Better than One: Combining Suggestions for Case Adaptation\*

David Leake and Joseph Kendall-Morwick

Computer Science Department, Indiana University, Lindley Hall 215  
150 S. Woodlawn Avenue, Bloomington, IN 47405, U.S.A.  
{leake, jmorwick}@cs.indiana.edu

**Abstract.** How to automate case adaptation is a classic problem for case-based reasoning. Given the difficulty of developing reliable case adaptation methods, it is appealing to consider methods which can exploit the strengths of a set of alternative adaptation methods. This paper presents a framework for combining suggestions from multiple adaptation methods, and illustrates and evaluates the approach in the context of interactive support for user modification of scientific workflows. The paper presents four adaptation methods for this domain, describes a method for assessing their confidence, proposes four strategies for suggestion combination, and evaluates the performance of the approach. The evaluation suggests that, for this domain, results depend more strongly on the adaptation methods chosen than on the specific combination method used, and that they depend especially strongly on a confidence threshold used for limiting irrelevant and incorrect suggestions.

## 1 Introduction

How to automate case adaptation is a classic problem for case-based reasoning. Numerous adaptation methods have been developed, requiring varying amounts of knowledge (see [1] for an overview). The use of “Knowledge light” approaches (e.g., [2]) facilitates the development of adaptation systems, but may not capture important aspects of the domain; knowledge-rich approaches are powerful, but the needed knowledge may be expensive to encode, and may have gaps if the knowledge engineer fails to correctly anticipate future adaptation problems.

Each case-based reasoning system normally relies on a single adaptation method, selected for the system based on the task characteristics and knowledge availability. Given the potential trade-offs of different adaptation methods, an interesting alternative is to endow each CBR system with a range of methods—potentially including a standard mix of knowledge-light methods with any available knowledge-rich methods—for the system to automatically assess the set of

---

\* This material is based on work supported by the National Science Foundation under Grant No. OCI-0721674. We thank Beth Plale and the Indiana University SDCI group at IU for their vital contributions to this work and the anonymous reviewers for their helpful comments.

suggestions and exploit whichever it expects to be most useful, given the task context and the relationships between the current suggestions.

This paper presents a framework for combining suggestions from multiple adaptation methods, and illustrates and evaluates the approach in the setting of an interactive system. The system provides incremental adaptation suggestions to a user who selects the suggestion to use, providing feedback to the system. The paper's examples consider the combination of methods at varying levels of domain independence; the methods for combination would be equally applicable to combining suggestions in a knowledge-rich adaptation setting.

Our approach bases combination decisions on both (1) confidence assessments for individual methods and (2) the context provided by the suggestions of other methods. Together, these are used to estimate confidence for the proposed adaptations. We present several methods for performing this combination task and evaluate them in a case study of Phala, an ongoing research project to develop a case-based system to assist scientific workflow generation [3].

This paper begins with a brief overview of Phala and its task domain. It then describes a set of individual adaptation methods and strategies for their combination. It closes with an evaluation examining the performance of the individual adaptations and of different combination methods. Because the usefulness of interactive systems also depends on how they balance suggestion quality against the frequency with which they present suggestions, the paper also evaluates how suggestion quality trades off against a user-specified suggestion rate.

Combining multiple adaptation methods relates to previous work on learning multiple local adaptation methods [4], but the approach presented here contrasts in considering multiple adaptation methods (which may change with experience) for each problem, rather than generating single methods for each region. The approach is in the spirit of the ensemble techniques which have been applied to CBR (e.g., [5]), but it emphasizes combining different types of methods which may not individually be capable of generating solutions for the entire space. This enables the system's starting set of adaptation methods to be chosen not for global characteristics, but for the degree to which the methods complement each other and fill specific knowledge gaps in the system.

## 2 Case Study Context: Intelligent Assistance for Authors of Scientific Workflows

### 2.1 Background

The scientific community is placing considerable emphasis on information technology to facilitate experimentation, and particularly on *e-Science*—*in silico* experimentation in which all experimental activities are simulated on a computer. Workflow technology plays a major role in supporting e-Science processing [6]. Workflows represent the control and data flow through experiments which may use a number of services for simulation, analysis, and transformation of data. For example, services might be selected to perform a specific simulation and then to analyze its results.

Workflows are represented by labeled directed multi-graphs in which nodes represent services and edges can represent either control or dataflow between these services, as well as input and output from the workflow itself. Scientists developing e-Science experiments must define the workflows for these experiments, but authoring a workflow can be a difficult task. The technology is often complicated, involving numerous steps, and authors are faced with many choices throughout the process. Consequently, a number of efforts have aimed to assist scientists in workflow generation, using both interactive and generative planning approaches (e.g., [7,8]).

The Phala project (described in detail in [3]) investigates case-based workflow generation support. Phala applies CBR to workflow author assistance by exploiting provenance information—cataloged traces of control and data flow from previous executions of individual workflows—as its source of cases. Phala is interactive, making suggestions to users for incremental improvements to partially completed workflows. Once a workflow is completed and executed, the resulting provenance is recaptured and included as a new case in the case-base. When a user builds a workflow, Phala uses the stored cases to incrementally suggest additions to the workflow, based on choices reflected in previous workflows. We refer to this process as case-based suggestion extraction. In addition to generating suggestions from past cases, Phala can generate suggestions based on global statistics for nodes likely to be connected in a workflow. We combined these methods into a hybrid method which outperformed each method individually [3]. The success of this combination method suggested investigating a more general combination approach.

### 3 Towards a General Framework for Combining Adaptation Suggestions

Two primary factors determine the performance of suggestion systems: The percentage of instances in which they make suggestions and the quality of the suggestions made. Managing both of these depends on the CBR system being able to assess confidence in the suggestions generated and to reconcile conflicting suggestion information. This requires addressing three main issues which apply to suggestion systems in general: (1) determining which of overlapping suggestions are relevant to the user's task, (2) determining levels of confidence in the relevant suggestions, (3) combining the advice of those suggestions.

#### 3.1 Determining Confidence in Candidate Suggestions

Confidence scores for suggestions can aid users as they select suggestions to apply, and can also facilitate internal choices by stand-alone systems. However, unless confidence can be normalized according to a uniform criterion, it may be difficult to compare confidence estimates for suggestions provided by different methods. In addition, it is possible that different suggesters could provide

both positive and negative information about a single suggestion (e.g., if a case-based suggester system has recorded a failure when the suggestion was applied previously, supporting a “contra-suggestion” that another suggester’s suggestion should not be followed). In the following sections, we illustrate how confidence estimates can be developed and how contra-suggestions can be integrated into the combination process.

### 3.2 Determining Relevance of Overlapping Suggestions

To facilitate discussion of overlapping suggestions, we first define some terminology to be used in the remainder of the paper. We refer to any aspect of a query for which an optimal suggestion strategy would produce a suggestion as an “opportunity” to make a suggestion. A query may contain many opportunities for suggestions to be made, and multiple suggestions may be returned for each query. For evaluation purposes, we assume that an oracle outside the suggester provides a set of desired suggestions, which we refer to as the “expected” suggestions.

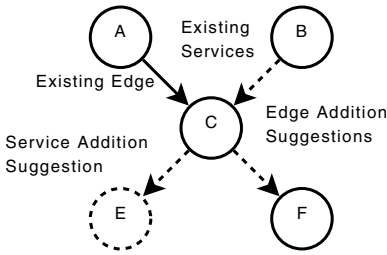
Which suggestions are relevant depends on the problem context in which the suggestions are generated. For example, for the workflow domain, we consider the problem context to consist of two components. The first is the type of suggestion (e.g., adding a service, adding an edge designating that output from one service is provided as input to another, deleting an edge, or replacing a service). The second is the location of the edit; specifically, the adjacent nodes for an added edge, deleted edge, or an altered or newly added node.

We classify suggestions as Irrelevant, Matching, Non-Matching, or Unused.

- **Irrelevant suggestions:** If the suggestion has no problem context in common with an expected suggestion, we consider it irrelevant (all other suggestions are “relevant”).
- **Matching suggestions:** For the purpose of evaluation, if both the problem context and the solution of a system-generated suggestion match an expected suggestion, then the system’s is “Matching”.
- **Non-Matching suggestions:** If the problem context of a system-generated suggestion overlaps with an expected suggestion’s but their solutions do not match, the suggestion is “Non-Matching.”
- **Unused suggestions:** When considering a set of mutually exclusive suggestions (i.e., suggestions for which adopting one suggestion would rule out adopting any of the others), the system first presents to the user the suggestion with the highest confidence. This is the only suggestion considered in the evaluation, with the remaining mutually exclusive suggestions considered “unused” rather than non-matching, regardless of whether or not they contain a matching suggestion. This way, the system is judged by its primary decisions, though in practice users may request to view the additional results.

Figure 1 illustrates problem context and matching in the workflow domain.





**Fig. 1.** Example suggestions for Phala

Suppose that an expected suggestion adds an edge from B to C, and that the system generates two edge addition suggestions, adding an edge from B to C and an edge from C to F. Both system suggestions would share problem context because they both include a node from the expected suggestion. However, only the edge suggestion from B to C is matching. An add service suggestion from C to E, despite being linked from the same node, is irrelevant because it is a different type of suggestion.

### 3.3 Combining Suggestions and Suggestion Confidence

Suggestions that partially or fully overlap may be combined to produce consistent and less confusing advice for the user. Likewise, if identical suggestions are generated from multiple suggesters, the overall confidence that the system has in those suggestions should reflect the individual confidence scores, requiring combination methods. The combination problem also presents issues such as how to deal with the confidence of mutually exclusive suggestions and how contra-suggestions should be combined with other suggestions.

If two independent suggesters generate the same suggestion, and each has high confidence in that suggestion, we might consider those independent confirmations to increase the overall confidence. However, if both suggesters base their suggestions on overlapping knowledge, the effect of the repeated suggestions on confidence is not as clear. The following section explores some combination methods reflecting this.

### 3.4 A General Framework for Producing and Combining Adaptation Suggestions

Given a set of suggestion methods, our framework combines their results by a three-step process. We introduce the general approach in this section and then examine it with a case study in the following section.

The steps in our framework are:

1. Generate suggestions using each component method
2. Critique suggestions among component methods
3. Combine suggestions

In the first step, each method is presented with an adaptation query for which it develops a set of suggestions (in our case, the query is a user request for suggestions of edits to a partially completed workflow). The resulting suggestions are then shared, with each method given the opportunity to review the current pool of candidate suggestions and either provide additional suggestions (these

may overlap with those offered by other methods) or contra-suggestions. For each suggestion and contra-suggestion, the method provides an estimate of confidence indicating the degree to which the component considers it a correct suggestion or not. Upon completion of the critiquing step, confidence scores are transformed to estimated probabilities of correctness and a combination method is applied to the estimated probabilities for the suggestions, resulting in a subset of the suggestions with new confidence values. Finally, resulting combined suggestions are presented to the user for approval and feedback. The feedback is captured for future case-based use either as additional suggestion support or to be used as a contra-suggestion.

We note that a similar framework was proposed in the context of information extraction by Freitag, who developed a scheme for combining multiple classifiers [9]. Freitag transforms confidence scores into correctness probabilities and uses these scores in one of three combination schemes (two of which are the basis for schemes described later in this paper). However, in addition to bringing such an approach to the adaptation combination task for CBR, our framework differs in elaborating on the issue of suggestion overlap and how it should be handled by general combination methods, in particular through the new contributions of suggestion review before combination and consideration of contra-suggestions, as well as the two new combination strategies described in the next section.

## 4 A Case Study of the Framework: Implementation in Phala

### 4.1 Component Suggestion Methods

Phala includes four suggestion strategies, each drawing on a different type of knowledge: (1) extraction of suggestions from provenance cases, (2) reuse of feedback cases, (3) link frequency, and (4) heuristics for the workflow domain.

*Extraction from Provenance Case.* This was the first case-based method developed for Phala [3]. This method draws from a case library which is mined from provenance traces of prior completed workflows. Cases with similar components to the query are retrieved, and workflow components in the retrieved case that do not exist in the query are extracted as potential additions to the query workflow. Suggestions extracted via this method do not represent complete solutions, but rather an incremental step in an interactive approach to generating an entire workflow. The final workflow is recaptured by the system when the completed workflow is executed and the resulting provenance trace is mined. This method uses single nearest neighbor retrieval, where similarity and confidence are both determined by edit distance from a stored workflow case to the query workflow.

*Reuse of Feedback Cases.* This is a second case based method, which generates suggestions from feedback cases rather than stored workflows. Whenever the system presents a suggestion to the user, the user has the option of providing

feedback as to whether it is a good suggestion. When feedback is received, a new case is created in the feedback case base, where cases are individual queries (incomplete workflows) rather than the complete workflows used by the Extraction method. This method uses 3-NN retrieval to retrieve prior solutions and re-presents them as potential solutions for the query. Confidence is determined by a combination of edit distance and prior confidence.

*Link Frequency.* This suggestion method was previously used as a baseline for evaluating the Extraction method [3]. For any service in the workflow, this method suggests adding the new service which is most often linked to (or from) the given service. The confidence is calculated as the ratio between the number of times the new service is linked with the given service and the number of times any service is linked with the given service in the cases of the case library.

*Workflow Heuristics.* This is our only method which relies on domain knowledge outside of the workflow case base and feedback case base. This method applies a collection of rules of thumb about workflow development. For example, a service within a workflow often takes a static number of inputs. Thus, in the critiquing round of combination, this method may suggest against an attempt to add more inputs than typical to a particular service. In this instance, the variance of the number of inputs is used to determine confidence.

## 4.2 Refitting to Insure Comparable Confidence Values

One solution to the problem of confidence comparability is to view confidence as a probability estimate. Frietag adopts such a strategy aiming to estimate the probability that a particular text fragment represents a field [9]. In order to produce these estimated probabilities, Frietag produces transformations for the confidence scores, mapping them to probabilities by producing a regression model from a set of mapped datapoints. Phala applies a similar approach, using an estimated probability that a suggestion will be matching or the probability that a contra-suggestion will be non-matching. These estimates are derived from confidence values using linear regression.

## 4.3 Methods for Combining Suggestions

We have developed a set of methods to combine suggestions by considering their problem context, solution, and confidence values. When a set of suggestions is presented to any of these combination methods, it is first partitioned by shared problem context. The following subsections describe how each method selects among one of these partitioned sets. Figure 2 provides algorithms for each method. In the algorithms,  $S$  is the relation which determines if two suggestions share a solution,  $M$  is the relation determining if suggestions share a source, and  $sugg$  is the set of all suggestions being presented to the combination method, partitioned by problem context.  $conf$ ,  $prob$ , and  $sol$  respectively represent the confidence, problem context, and solution for a particular suggestion.

```

while |sugg| > 0 do
  if conf(max(sugg)) < 0 then
    sugg ← sugg ∩ {si ∈ sugg : sMax(sugg)}
  else
    return max(sugg)
  end if
end while
return ∅

```

(a) The Maximum Combiner

```

ret ← ∅
for all suggi such that suggi ∈ Partition(sugg, S) do
  conf(si) ← ∑s ∈ suggi  $\frac{\text{conf}(s)}{|sugg_i|}$  for some si ∈ suggi
  ret ← ret ∪ {si}
end for
return Maximum(ret)

```

(b) The Average Combiner

```

ret ← ∅
for all suggi such that suggi ∈ Partition(sugg, S) do
  sugg+ ← {s ∈ suggi : conf(s) > 0}
  sugg- ← {s ∈ suggi : conf(s) < 0}
  P ← 1 - ( ∏s ∈ sugg+ 1 - conf(s) )
  N ← ∏s ∈ sugg- 1 - |conf(s)|
  conf(si) ← P × N for some si ∈ suggi
  ret ← ret ∪ {si}
end for
return Maximum(ret)

```

(c) The Independent Combiner

```

ret ← ∅
for all suggi such that suggi ∈ Partition(sugg, M) do
  if |Partition(sugg, M)| = 1 ∧ chatty(Average(suggi)) then
    return ∅
  end if
  ret ← ret ∪ {Average(suggi)}
  ret ← ret ∪ {s ∈ suggi : conf(s) < 0}
end for
return Independent(ret)

```

(d) The Balanced Combiner

**Fig. 2.** Combination Methods

*Maximum.* The suggestion with the highest confidence score is chosen. If this is a contra-suggestion, all suggestions with matching solutions are removed from the set and the process is repeated. If no suggestions remain, none is returned. This is based on the CMax method used by Freitag [9].

*Average.* This method first partitions suggestions by shared solution. The confidence scores for each of these partitions are averaged (confidence scores for contra-suggestions are negative). If the average is negative, the suggestion is switched to a contra-suggestion. The resulting suggestions are then passed through the Maximum combination method.

*Independent.* This method also first partitions suggestions by shared solution. Under the assumption that each confidence score is independent, the probability that a solution is correct is decided by calculating the probability that all matching positive suggestions are not wrong and that all matching contra-suggestions are wrong. For each partition, one solution is returned with this recalculated probability score. These results are then passed through the Maximum combination method. This is derived from the CProb method used by Freitag which calculates the probability that at least one prediction for a particular solution is correct [9].

Note that a problem with the Independent combination method is that confidence scores are not always independent. Although, ideally, suggesters will complement each other, in practice some knowledge may be shared by the suggesters.

*Balanced.* Suggesters are not restricted from making multiple suggestions with the same problem context or even the same solution. This is not necessarily

incorrect. For example, the Extraction suggester might make 6 suggestions to add the same service in a particular location, because in the stored case 6 instances of that service were executed in parallel. However, the Independent combination method would consider each of these to be independent support, potentially causing the Independent suggester to prefer it over better individual solutions provided by other Suggesters. The Balanced combiner addresses this problem by first partitioning suggestions by which method was used to generate them and reducing each partitioned set to a single suggestion by using the Average combiner.

Some suggesters could produce many irrelevant suggestions. For example, Link Frequency is ignorant of solution relevance, and consequently makes many irrelevant suggestions. These suggestions are unlikely to be supported by other suggestion methods. Consequently, the Balanced combiner also reduces irrelevant suggestions by ignoring suggestions which (1) lack the support of other methods, (2) have low confidence scores and (3) are produced by methods known to create many irrelevant suggestions. This is done by requiring each suggestion's confidence to surpass a specific confidence threshold in order to be considered, unless it is supported by other methods. After this and the previous step, all remaining suggestions are combined with the Independent combiner.

## 5 Experimental Design

### 5.1 General Factors and Trade-Offs

Ideally, a combiner would maximize matching suggestions and minimize non-matching and irrelevant suggestions. We shall consider the ratio of matching suggestions to opportunities as the “matching rate” and the ratio of irrelevant suggestions to opportunities as the “irrelevance rate”. In Phala, irrelevant suggestions are only generated when the user does not identify a problem context, leaving the system to guess where suggestions are most applicable. These are the only scenarios we consider in our evaluation. We also identify the ratio of relevant suggestions to opportunities as the “suggestion rate”, ideally maximized as well, however, not at the expense of either of the prior listed goals. We use the term matching rate instead of precision because we cannot know if alternative suggestions may actually be relevant. A user's previous choices are not a complete representation of potential relevance. Thus, the matching rate is a lower bound on precision. Likewise, the irrelevance rate is an upper bound on true irrelevance.

An additional factor is a user-selected threshold for the the minimum confidence a suggestion can have, which can be used to limit the number of non-matching and irrelevant suggestions presented. Our evaluation also examines how the above rates change when a minimum confidence threshold is enforced. For instance, a method with a matching rate of 30% might have a 70% matching rate with a higher confidence threshold, though this necessarily means that it will have a lower suggestion rate as well. In order to take this into account, we

graph matching and irrelevance rates against suggestion rate. These values can be found by adjusting the minimum confidence threshold until a desired suggestion rate is achieved. Note that methods typically have a maximum suggestion rate below 1.0, which means that the graphs we produce will not be defined at all points.

## 5.2 Dataset

Our evaluation of the alternative methods is performed with a current snapshot of the 447 public workflows available at [myexperiment.org](http://myexperiment.org) [10]. This collection focuses mainly on the bio-sciences, but includes a diverse contribution of workflows from other disciplines as well. These workflows are used to build up the provenance case-base and also as queries through leave-one-out testing.

## 5.3 Leave-One-Out Tests

The evaluation is based on a series of unique leave-one-out tests which test Phala through several different suggestion scenarios. In each test, one workflow is withheld from the case-base and altered in some way. Phala is then queried and the expected suggestions should return the workflow to its previous state. The tests are:

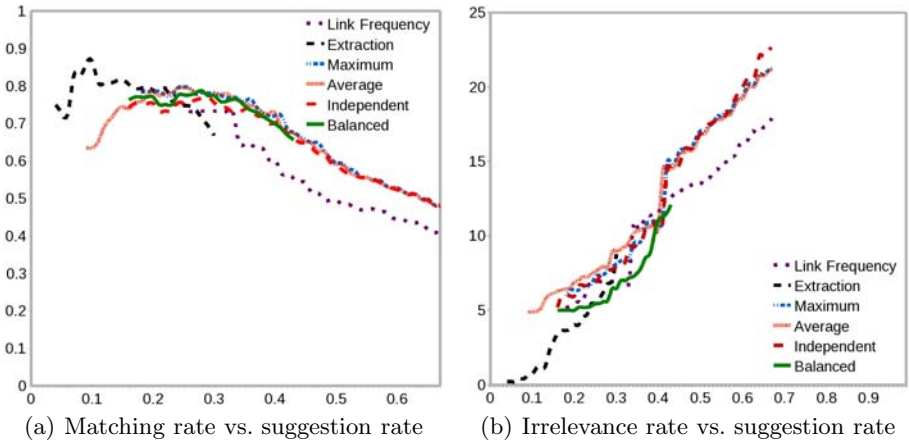
- **Service Addition Test:** One service is deleted and must then be suggested as an addition.
- **Service Replacement Test:** One service is replaced with an erroneous service and the original must be suggested as a replacement.
- **Edge Addition Test:** One edge is deleted and must then be suggested as an addition.
- **Edge Deletion Test:** An erroneous edge is added and a suggestion to delete it must then be made.
- **Reconstruction Test:** The reconstruction test rebuilds a workflow from its input and output components through multiple incremental suggestion sessions. This is intended to model a more realistic user session.

## 6 Results

Results from each constituent method and each combiner method for the service addition test are listed in Table 1 and results for each combiner method for the remaining tests are listed in Table 2. For a 95 % confidence Z-test, Maximum, Average, and Independent combination methods do not significantly differ in matching or suggestion rates. They also do not significantly differ in irrelevance rates except between Independent and either Maximum or Average for the service addition test. Balanced differs significantly from these combiners for all rates in all cases. Given that Balanced has a nearly identical matching rate with

**Table 1.** Each method’s performance for the Add Service test (1638 Opportunities) (*note: Both Workflow-Heuristics and Feedback produce many contra-suggestions which are not included in this table but are important to the success of the combiner methods*)

Method	Matching (Rate)	Non-Matching	Relevant (Rate)	Irrelevant (Rate)
Extraction	331 (65.7%)	173	504 (30.8%)	14990 (9.15)
Feedback	12 (85.7%)	2	14 (0.85%)	28 (0.017)
Link Frequency	450 (40.7%)	655	1105 (67.5%)	30199 (18.43)
Maximum	531 (48.1%)	573	1104 (67.4%)	36023 (22.0)
Average	529 (47.9%)	575	1104 (67.4%)	36023 (22.0)
Independent	528 (47.7%)	577	1105 (67.5%)	41176 (25.1)
Balanced	463 (65.2%)	247	710 (43.4%)	20125 (12.3)



**Fig. 3.** Service Addition Test Performance

Extraction, a significantly higher suggestion rate, and a significantly lower irrelevance rate, these data suggest using one of these two methods. If the suggestion rate is important to the user, Balanced has a clear advantage in our tests.

However, this does not take into account a user’s ability to specify a minimum confidence threshold for suggestions returned by the system. Because each of the other combination methods have higher suggestion rates than Balanced, it is possible that they would also have a higher matching rate were the confidence threshold raised to the point that their suggestion rate equaled that of Balanced in the table above. To investigate this possibility, we graphed matching rate against suggestion rate for the combiners and Extraction and Link Frequency constituent methods in Figure 3(a). This figure shows that the combiners make many more suggestions at the point at which they are most accurate than either of the constituents. At these points, both of the constituent methods are less accurate and continue to be for all scenarios with higher suggestion rates. Thus the combiners produce superior results. Though

irrelevance rates are fairly high for all methods in Tables 1 and 2, this graph also illustrates how the user can significantly lower the irrelevance rate to a level which may be more acceptable by increasing the minimum quality threshold.

Also notable in this graph is that each combiner method shares similar performance regardless of suggestion rate (with the exception of Balanced having a smaller upper bound on possible suggestion rates). This was a surprising result as the derived combiners were less consistent in Freitag’s work [9]. In Figure 3(b), irrelevance rates are graphed for each of the methods, showing that in this testing situation, the Balanced combiner fares better than the other combiners through nearly all of its higher suggestion rates. In this testing scenario, unless a very high suggestion rate is desired, the Balanced combiner seems to be the best compromise.

However, this trend was not observed on all tests. The irrelevance rates for the combiner methods on the reconstruction test are listed in Figure 4. Each method again shared similar matching rates in this test, but Balanced no longer has a superior irrelevance rate. Although Balanced was more successful in other scenarios, it seems that, at least for this particular system, the most reliable factor in reducing unwanted or incorrect suggestions is not the selection of combiner method, but rather the proper management of the confidence threshold.

**Table 2.** Combiner performance for various tests

Method	Matching (Rate)	Non-Matching	Relevant (Rate)	Irrelevant (Rate)
<b>Reconstruction Test (2620 Opportunities)</b>				
Maximum	379 (48.9%)	396	775 (29.6%)	42243 (16.12)
Average	380 (49.1%)	393	773 (29.5%)	42249 (16.13)
Independent	375 (46.6%)	429	804 (30.7%)	44952 (17.16)
Balanced	340 (58.1%)	245	585 (22.3%)	24482 (9.34)
<b>Edge Suggestion Test (1638 Opportunities)</b>				
Maximum	379 (78.1%)	106	485 (29.6%)	37837 (23.10)
Average	379 (78.3%)	105	484 (29.5%)	37840 (23.10)
Independent	387 (71.0%)	158	545 (33.3%)	42882 (26.18)
Balanced	377 (77.7%)	108	485 (29.6%)	20501 (12.52)
<b>Edge Deletion Test (1994 Opportunities)</b>				
Maximum	648 (84.7%)	117	765 (38.4%)	24711(12.39)
Average	639 (83.7%)	124	763 (38.3%)	24712 (12.39)
Independent	648 (84.7%)	117	765 (38.4%)	28065 (14.07)
Balanced	644 (84.2%)	121	765 (38.4%)	13700 (6.87)
<b>Service Replacement Test (1852 Opportunities)</b>				
Maximum	320 (52.1%)	294	614 (33.2%)	37558 (20.28)
Average	319 (52.0%)	295	614 (33.2%)	37558 (20.28)
Independent	319 (52.0%)	295	614 (33.2%)	43206 (23.33)
Balanced	316 (67.2%)	154	470 (25.4%)	20595 (11.12)



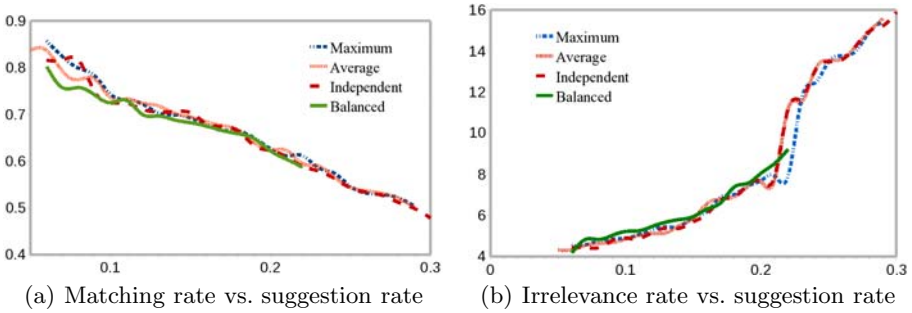


Fig. 4. Reconstruction Test Performance

## 7 Related Work

The most salient aspect of our framework is its general approach to hybridization. Related CBR work includes the ensemble techniques such as those by Craw et al. [11] and Plaza et al. [5], though Plaza et al.’s study does not use distinct methods between agents, only distinct knowledge sources. A similar distinction can be made with multi-case based reasoning [12]. The cited ensemble techniques approach the issue of combining solutions through voting schemes which are more applicable to solution representations which lack a notion of “problem context”.

Our framework is more similar to previously cited work in Information Extraction from which some of our methods are derived [9], though we have expanded the information content of solutions and developed new combination methods. Other information extraction work expanding on Freitag’s makes relevant contributions to solution combination and also utilizes “merged templates” which is an example of partitioning by problem context [13]. Work in boosting is also relevant [14].

There is also work related to the domain and the individual suggesters utilized in Phala, much of which is noted in a previous paper [3]. There has been work on mining adaptation knowledge from a case-base, which is a strategy taken by the Workflow Heuristics and Link Frequency methods (e.g., [15]). The Feedback method utilizes a case base of acquired knowledge for adaptation in the spirit of work by Leake, Kinley, and Wilson [16]. This work also relates to recent interest in case-based confidence (e.g., [17]), but differs in that it concerns confidence not in cases, but in candidate adaptations to apply to them.

## 8 Future Work

Our results illustrate the importance in maintaining a proper confidence threshold to meet user goals. This being the case, we hope to develop methods for automating setting of this and other relevant system parameters so that the user is not required to experiment in order to achieve desired results. Because of the importance of confidence values, we also seek to improve the confidence

translation. Regression in this context is not straight-forward because data points must be derived from a proportion of binary values. We hope to achieve a better solution than those offered in the literature or in our current implementation, though we also expect that a larger training set would yield better performance even with the current methods. We also intend to explore additional suggesters, including more knowledge-rich methods (such as employing an ontology or user generated semantic tagging).

In our current implementation we consider problem contexts and solutions to be only matching or non-matching, when in fact these entities may partially match. This raises a host of issues, from implementation of combination methods to performing a suitable evaluation, which we hope to address in future work. In addition, we would like to expand our notion of problem contexts to facilitate discovery and exploitation of particular contexts in which particular methods may excel.

## 9 Conclusions

In conclusion, our framework for combining suggestions is a promising means for facilitating adaptation in interactive CBR systems, yielding results superior to the constituent methods. This framework relies on effective confidence scores which are important for combining suggestions. We show that the confidence scores can also be used to select more desirable results and improve overall system performance by meeting specific user preferences on various trade-offs in performance.

We have demonstrated that this framework as well as several new constituent methods can be effectively applied to the domain of supporting the creation of scientific workflows. We have found that the choice and number of these methods had a greater impact on the results than the method of combination, in contrast to previous results on combiners for classification. We see techniques we have presented as applicable to other approaches within our domain as well as to problems in other domains and plan to investigate this potential in future work.

## References

1. Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision, and retention in CBR. *Knowledge Engineering Review* 20(3) (2005)
2. Wilke, W., Vollrath, I., Althoff, K.D., Bergmann, R.: A framework for learning adaptation knowledge based on knowledge light approaches. In: *Proceedings of the Fifth German Workshop on Case-Based Reasoning*, pp. 235–242 (1997)
3. Leake, D., Kendall-Morwick, J.: Towards case-based support for e-science workflow generation by mining provenance information. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) *ECCBR 2008*. LNCS, vol. 5239, pp. 269–283. Springer, Heidelberg (2008)
4. Patterson, D., Rooney, N., Galushka, M.: A regression based adaptation strategy for case-based reasoning. In: *Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence*, pp. 87–92. AAAI Press, Menlo Park (2002)

5. Plaza, E., Ontañón, S.: Ensemble case-based reasoning: Collaboration policies for multiagent cooperative CBR. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS, vol. 2080, p. 437. Springer, Heidelberg (2001)
6. Yu, J., Buyya, R.: A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.* 34(3), 44–49 (2005)
7. Xiang, X., Madey, G.R.: Improving the reuse of scientific workflows and their by-products. In: ICWS, pp. 792–799. IEEE Computer Society, Los Alamitos (2007)
8. Maechling, P., Chalupsky, H., Dougherty, M., Deelman, E., Gil, Y., Gullapalli, S., Gupta, V., Kesselman, C., Kim, J., Mehta, G., Mendenhall, B., Russ, T., Singh, G., Spraragen, M., Staples, G., Vahi, K.: Simplifying construction of complex workflows for non-expert users of the southern california earthquake center community modeling environment. *SIGMOD Rec.* 34(3), 24–30 (2005)
9. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. PhD thesis, Carnegie Mellon University (1998)
10. Roure, D.D., Goble, C., Bhagat, J., Cruickshank, D., Goderis, A., Michaelides, D., Newman, D.: Myexperiment: Defining the social virtual research environment. In: 4th IEEE International Conference on e-Science (August 2008)
11. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence* 170(16–17), 1175–1192 (2006)
12. Sooriamurthi, R.: Multi-case-base reasoning. PhD thesis, Indiana University (2007)
13. Sigletos, G., Paliouras, G., Spyropoulos, C.D., Hatzopoulos, M.: Combining information extraction systems using voting and stacked generalization. *J. Mach. Learn. Res.* 6, 1751–1782 (2005)
14. Dietterich, T.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
15. d’Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 750–755. Morgan Kaufmann, San Mateo (2007)
16. Leake, D., Kinley, A., Wilson, D.: Learning to improve case adaptation by introspective reasoning and CBR. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 229–240. Springer, Heidelberg (1995)
17. Cheetham, W., Price, J.: Measures of solution accuracy in case-based reasoning systems. In: Funk, P., González, P. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 106–118. Springer, Heidelberg (2004)

# Adaptation versus Retrieval Trade-Off Revisited: An Analysis of Boundary Conditions

Stephen Lee-Urban and Héctor Muñoz-Avila

Department of Computer Science and Engineering, 19 Memorial Drive West,  
Lehigh University, Bethlehem, PA 18015  
{sml3, hem4}@lehigh.edu

**Abstract.** In this paper we revisit the trade-off between adaptation and retrieval effort traditionally held as a principle in case-based reasoning. This principle states that the time needed for adaptation reduces with the time spent searching for an adequate case to be retrieved. In particular, if very little time is spent in retrieval, the adaptation effort will be high. Correspondingly, if the retrieval effort is high, the adaptation effort is low. We analyzed this principle in two boundary conditions: (1) when very bad and (2) when highly capable adaptation procedures are used. We conclude that in the first boundary condition the adaptation-retrieval trade-off does not necessarily exist. We also claim that the second does not hold for a class of planning domains frequently used in the literature. To validate this claim, we performed experiments on two domains of this type.

**Keywords:** cased-based reasoning, planning, retrieval, adaptation, trade-offs.

## 1 Introduction

One of the crucial principles of case-based reasoning is the trade-off between the retrieval time, the time it takes to find a relevant case for a given problem from the case base, and the adaptation time, the time it takes to adapt the retrieved case. The trade-off has been summarized in Fig. 1, taken from [1]. There are three tenets of this principle:

1. If little time is spent on retrieval, then, on average, the adaptation effort involved in using the retrieved cases to solve the given problem will be high. This is basically the result of stopping the retrieval process too early, which results in the retrieval of cases that are not easy to adapt.
2. If too much time is spent on retrieval, then the adaptation effort to solve the given problem will be small. This is basically the result of spending enough time to ensure that a case is retrieved that is easier to adapt. However, any reduction in the adaptation time is counterbalanced by the time spent in retrieval, which may result in a high overall problem-solving time (adding up retrieval and adaptation times).

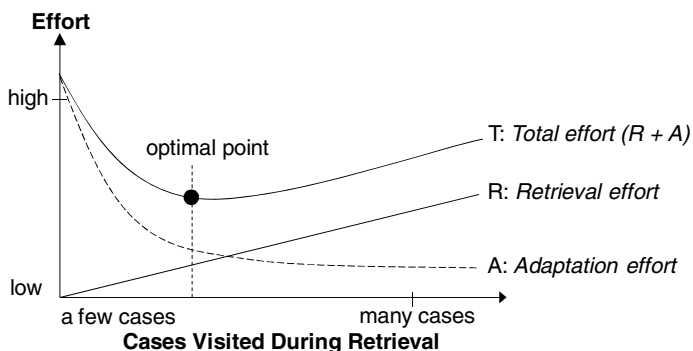


Fig. 1. Adaptation and retrieval trade-off (Veloso, 1994)

3. There is an optimal intermediate point at which a case that is good enough to adapt is retrieved and adapting it produces the smallest overall time.

The tenants of this principle have been recurrently discussed over the years. It was in part the motivation for retrieval strategies in case-based planning systems such as CAPLAN/CbC [2], derSNLP+EBL [3], and MRL [4]. It was also observed by studies about the occurrence of the utility problem in case-based reasoning [5]. It continues to be encountered in a number of domains including software design [6], travel domain [7], and process planning [8].

Improvements in retrieval procedures (e.g., [9], [10]) can be seen as an effort to reduce the time to find cases “good enough” for fruitful adaptation. Analogously, improvements in adaptation (e.g., [11], [12], [13], [14], [15]) can be seen as indirectly reducing the time needed for retrieval because as adaptation is improved, less time need be spent on finding “good enough” cases to be adapted. Hence both kinds of improvements, in adaptation and retrieval, can be seen as moving the optimal point depicted in Fig. 1, and described in the third tenet, downward, making the overall problem-solving effort less costly.

The rest of the paper proceeds as follows: In Section 2 presents our analysis of the two boundary conditions. Section 3 provides an illustrative plan adaptation example. In Section 4, we describe domain-configurable plan adaptation by first reviewing partial-order planning (4.1), then by explaining the domain-configurable plan adaptation knowledge (4.2) used by our adaptation procedure (4.3). Next, Section 5 gives an example of domain-configurable plan adaptation. The details of our empirical evaluation are presented in Section 6, followed by concluding remarks.

## 2 Analyses of Boundary Conditions

An implicit assumption in the adaptation-retrieval trade-off principle is that the adaptation algorithm is only capable of solving problems quickly enough when the retrieved case is reasonably similar to the input problem. If the retrieved case is not sufficiently similar to the new problem then the adaptation effort will take a significant amount of time. It is worth noting that despite advances in adaptation algorithms, there is still a search process needed in a potentially exponential search space. The

relationship between this search space and techniques for its exploration can be loosely summarized as follows: A case sufficiently similar to the input problem is retrieved and adjusted in a sensible way. The output of this process is an “adjusted plan” where the retrieved plan is partially adapted but is an incomplete solution to the input problem. Hence, further search is required to reach a solution node of the new problem starting from the adjusted plan. This search can be performed either by first-principles search as in [11] or by retrieving and adapting another case as in [12] or by composing multiple planning cases as in [16] or by combining further retrieval/adaptation of cases and first-principles search as in [1] and [17].

To make this assumption explicit, we will study two boundary cases: when the adaptation algorithm is capable of solving only those problems for which it has already stored solutions in the case base (naïve) and when a highly capable adaptation procedure (omniscient) is used.

### 2.1 Analysis of a Naive Adaptation Algorithm

Fig. 2 (left) illustrates the situation of a naïve adaptation procedure, capable of solving only those problems for which it already has a solution. The dotted segments represent discontinuities in problem-solving time, reflecting those problems for which a solution does not exist in the case base (CB) and therefore no data point can be drawn. The adaptation time for those problems for which a solution is already stored is zero because the solution is taken as-is. Consequently, the linear search of the retrieval procedure (as in Fig. 1, assume a constant time to compute similarity between case and problem, and a sequential search through the cases), yields an overall linear time for problem solving. Basically, the problem-solving time is the time taken for the retrieval procedure to find the solution in the case base, if one is already stored. This is an analogy of the CB working as a sequential database. In this situation there is no trade-off between adaptation and retrieval. The retrieval mechanism must continue looking for a solution until it finds an exact match or it has exhausted the whole case base.

### 2.2 Analysis of an Omniscient Adaptation Algorithm

More interesting and difficult is to analyze the situation where an omniscient adaptation algorithm is given. First, we would like to characterize such an algorithm. An

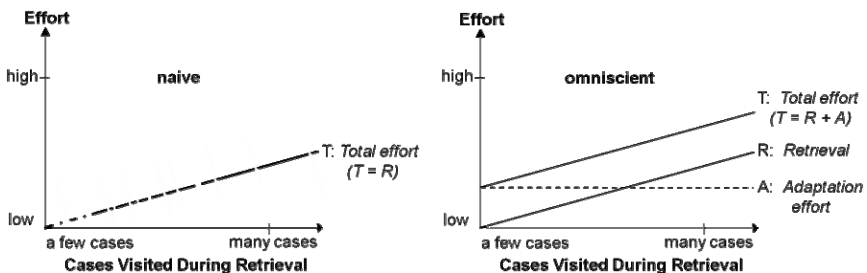


Fig. 2. Naive (left) and omniscient (right) adaptation algorithms without a trade-off

omniscient adaptation procedure is one where it would not need to backtrack to reach the adjusted plan nor backtrack during search to further refine the adjusted plan into a complete solution of the new problem. That is, somehow the algorithm finds the path in the search space to first reach the adjusted plan and then reach the solution plan without needing to explore alternative branches in the search space.

Our hypothesis is that there exists a class of planning domains such that the time it takes an omniscient adaptation algorithm to adapt any two cases to solve a new problem is roughly the same, regardless of the similarity of the individual cases to the new problem, and given that the problems solved by the cases and the new problem are of the same size (i.e., they have the same number of objects in their initial states). Once again, combined with the linear search of the retrieval procedure, this hypothesis gives an overall linear time for problem solving. Basically, the problem-solving time is proportional to the time it takes for the retrieval procedure to retrieve a case. We claim that in this situation there is no trade-off between adaptation and retrieval as illustrated in Fig. 2 (right).

This class of planning domains is one where the graph consisting of all world states is a directed, strongly connected graph. In this graph, the vertexes are world states and the directed edges are actions (i.e., an edge from vertex  $x$  to vertex  $y$  represents the action transforming the state  $x$  into state  $y$ ). We called these **connected domains**. In a connected domain, there always exists a directed path (sequence of actions) between any two vertexes (states) in the graph. The logistics transportation domain [1] is an example of a connected domain, provided that there is at least one truck and one airport in each city, as well as at least one airplane. With these provisions, a package in any location can always be relocated into any other location. Similarly, the blocks world [18] also meets this property: any configuration of blocks can always be reconfigured into any other configuration of these same blocks. An example of a domain that does not meet this property is a logistics domain variant where there are one-way routes between locations [3]. As a result, a package in a certain location may not always be reachable by a truck.

An important question is whether it is possible to construct an omniscient algorithm for connected domains. This is particularly compelling, considering that many experiments designed to demonstrate efficiency gains of new case-based planning techniques use connected domains. Additionally, these domains have an exponential search space and, hence, the question of whether adaptation procedures could be built that somehow adapt the retrieved plans without exploring unnecessary branches in the search space is a good one.

In this paper we will report on an omniscient adaptation algorithm, DCPOP-A (for: domain-configurable partial-order plan adapter). In Section 6, we report the results of experiments that demonstrate, for the connected domains described above, the non-existence of a trade-off between adaptation and retrieval in our omniscient adaptation algorithm as depicted in Fig. 2 (right).

### 3 A Plan Adaptation Example

Fig. 3 illustrates a snippet of the search space for state-space planning in blocks world. It shows 15 states (five of them labeled  $p1$ ,  $state\ 1$ ,  $state\ 2$ ,  $state\ 3$  and  $final$ )

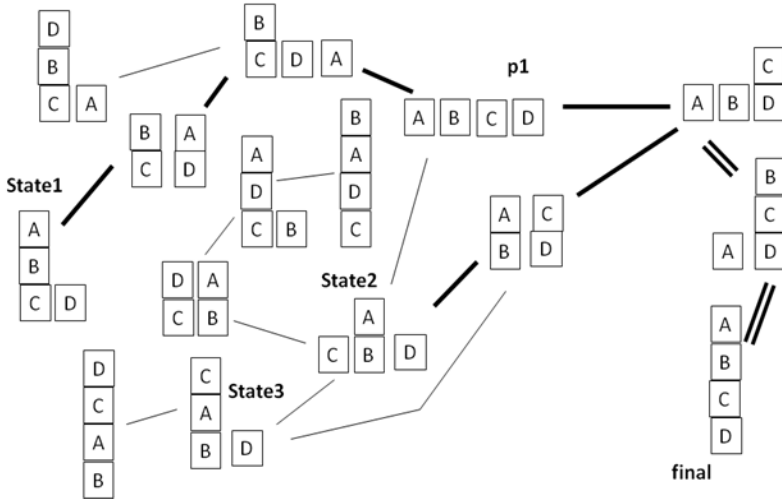


Fig. 3. Snippet of the search space with 4 blocks

out of the 73 possible states with 4 blocks. Lines connecting states represent transitions by the only action in the domain: *Move(?x ?y ?z)*. This action puts a clear block *?x* (i.e., a block with no other block on top of it) currently on top of a block *?y* (or on the table; *?y = table*) on top of another clear block *?z* (or on the table).

Transitions do not have a direction because with a change of parameters and/or actions they can go in either direction. Problems can be simply defined as pairs of states (*s*, *s'*) where *s* is the initial state and *s'* is the goal state. Suppose that two cases are stored in the case base, Case 1 and Case 2. Case 1 solves the problem (*state 1*, *final*) and Case 2 solves the problem (*state 2*, *final*). Their solutions follow the path between these two states denoted by the bold line-connectors (the double-line parts represent common portion of the solutions). Suppose that a new problem (*state 3*, *final*) is given. Of these two cases, Case 2 is a much better choice. In fact if starting from *state 3*, there are only 3 possible transitions, two of which lead directly to the solution plan of Case 2 and the third one (the 4 block pile) which is a dead-end. Case 1 might not be a good choice. For example, from *state 3* a path of length 4 leading to the dead-end representing the 4-block pile [B, A, D, C] can be explored. There is also a length 2 path to *p1* which would allow it to use the solution path of Case 1 but a first-principles search might take a significant amount of time before this path is found because it may first explore dead-end paths.

### 4 Domain-Configurable Plan Adaptation

Our approach for plan adaptation is motivated by existing research in domain-configurable planning. In this form of planning, domain-specific knowledge enhancing the action schemas is given. This knowledge is used to guide the planning process, which like first-principles planning generates a plan from scratch. Domain-configurable planners have been shown to solve problems more quickly and to scale much better with



problem size than first-principles planners. Because of their scalability, their increasing number of applications, and their ability to drop classical planning assumptions, domain-configurable approaches are believed to be closing the gap between academic research in AI planning and real-world applications [19].

We developed DCPOP-A, a domain-configurable plan adaptation algorithm, in order to investigate the adaptation-retrieval trade-off in a system capable of performing “omniscient” search with ideal inputs. This new problem-solving paradigm for plan adaptation uses domain-specific knowledge to guide a domain-independent plan adaptation process. The domain-independent property allows the semantics of the resulting planning algorithms to be clear. Domain-specific knowledge allows problem-solving to scale well with problem size. This, in addition to previous analyses of the search space, illustrates the potential for substantial speed-up gains in the plan adaptation process, thus providing a suitable framework in which to re-evaluate the adaptation-retrieval trade-off.

We used partial-order planning (POP) as the underlying planning formalism to conduct our research. POP was the dominant planning paradigm some 15 years ago because of its ability to flexibly interleave actions, rather than totally order them, while solving problems. POP drops the classical requirement for actions to be totally ordered, which is particularly useful for plan adaptation (e.g., [11], [12]). However, interest in POP waned when other paradigms such as analysis of planning graphs and more recently planning with heuristics, demonstrated significant gains in planning speed and solvable problem size. More recently, there has been a revival of POP as heuristic methods have been developed that perform comparably to other state-of-the-art first-principles planners. Researchers have pointed out the importance of POP planning for real-world domains because in many real world situations actions can be performed in parallel and the planner should not commit to step orderings unless necessary (e.g., [20], [21], [22], [23]).

#### 4.1 Partial-Order Planning

Partial-order planning begins with an input action schema and a symbolic initial and goal state specification of the problem. *Actions* have a name, zero or more parameters, preconditions, and effects. Next an *initial plan* is created, consisting of two special steps. The first of these steps has as effects those atoms appearing in the problem’s initial state; the second has as preconditions those atoms appearing in the goal state. Partial-order planning refines this initial plan by adding constraints and plan steps, ordered between the two initial steps, until a complete partial-order plan is obtained (complete plans are defined below). A *partial-order plan* is defined as a 4-tuple  $(S, \rightarrow, \rightarrow_{CL}, B)$  of sets of *POP plan elements*.  $S$  is the set of plan *steps*, which represent the application of actions in the plan. The set  $\rightarrow$  contains the *ordering constraints* between plan steps, which take the form  $s \rightarrow s'$ , indicating that step  $s$  must be executed before step  $s'$ . The set  $\rightarrow_{CL}$  contains *causal links*,  $s \rightarrow_p s'$ , indicating that the precondition  $p$  needed by the action in step  $s'$  is produced as an effect of the action in step  $s$ . Step  $s$  might be an existing step in the plan or a new one added to satisfy  $p$ . The set  $B$  indicates variable *binding constraints*,  $?x \neq ?y$  or  $?x = ?y$ , indicating that whenever variable  $?x$  occurs in the plan it must take a different (respectively the same) value as the variable  $?y$  ( $?x$  represents that  $x$  is a variable symbol). Set  $B$  is

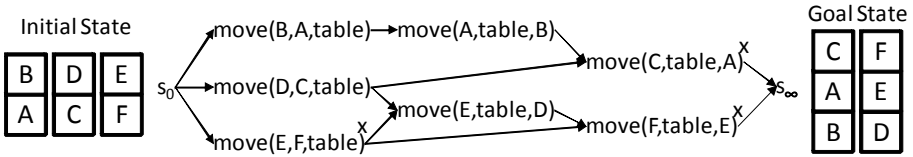


Fig. 4. Example of a partial-order plan

empty when planning without variables (i.e. “grounded”). A partial-order plan is **complete** if it has no **flaws**. There are two kinds of **flaws** in POP: open preconditions and causal threats. An **open precondition** occurs when a step  $s'$  in the plan has a precondition  $p$ , written  $p@ s'$ , for which no causal link  $s \rightarrow_p s'$  exists. A **threat** occurs when a causal link  $s \rightarrow_p s'$  and a step  $s''$  exist such that  $s''$  has as effect the negation of  $p$  (i.e.,  $\neg p$ ), written  $s'' \rightarrow_{\neg p}$ , and  $s''$  can occur between  $s$  and  $s'$ , written  $s'' \parallel (s \rightarrow_p s')$ , in a linearization of the plan. A **linearization** of a plan is a sequencing of all steps in a manner consistent with the ordering constraints such that, for every two steps  $s$  and  $s'$ ,  $s$  will always be listed before  $s'$  if either  $s \rightarrow_p s'$  or  $s \rightarrow_{\neg p} s'$  hold. The objective of the POP planning process is to refine the initial plan into a complete partially-ordered plan. Any linearization of this complete partially-ordered plan is a solution to the planning problem. There are four possible **POP plan refinements**: adding ordering constraints, adding steps, adding causal links, and adding binding constraints. Ordering constraints and binding constraints are added to solve causal threats. Steps and causal links are added to satisfy open preconditions.

Fig. 4 shows an example of a partial-order plan in the Blocks World domain. The arrows represent causal links and ordering constraints. The meaning of the “x” beside some of the plan steps will be explained in a later discussion and may be ignored for now. If nothing is underneath a block, this means that the block is on the table (not shown), For example, in the initial state block C is on the table and in the goals block B is on the table. This plan unstacks all blocks on the table and then stacks them to form the configuration indicated by the goals.

### 4.2 Domain-Configurable Partial-Order Plan Adaptation Knowledge

Domain-configurable partial-order knowledge in DCPOP-A is encoded as rules of the following form:

**if** (+/-) <POP plan element> [, (+/-) <POP plan element>]  
**then** (do:/undo:) <POP plan refinement> [, (do:/undo:) <POP plan refinement>]

The conditional part of the rule is a conjunction of one or more POP plan elements as defined in the previous section. These POP plan elements are preceded by either a plus or a minus. The consequent part is a sequence of POP plan elements, preceded by do or undo symbols. The semantics of a rule are as follows. The rule is satisfied if each of the POP plan elements preceded by a plus sign occurs in the current plan and none of the POP plan elements preceded by the minus sign occur in the current plan. The consequent part indicates each of the POP plan refinements to add, if it is preceded by a *do*, or to delete, if it is preceded by an *undo*. When a plan step  $s$  is deleted (i.e., by an *undo*), any ordering constraint or causal link connecting to/from  $s$  is also

removed. When an ordering constraint, a causal link, or a binding constraint is deleted, no other plan element is removed. The POP domain-configurable rules, which henceforth we refer to as POP rules, are a natural extension of POP refinements and in fact all POP refinements can be expressed using these rules. We also allow as conditions *same*(? $x$ ,? $y$ ) and *different*(? $x$ ,? $y$ ) indicating that two variables take the same (or different) value. We write instead  $?x = ?y$  (or  $?x \neq ?y$ ) for readability.

We use two classes of POP rules: retraction rules and refinement rules. **Retraction rules** indicate POP plan elements that must be removed from the plan. As a result, they always have the *undo*: label in the consequent part of the rule. **Refinement rules** indicate POP plan elements that must be added to the plan. As a result, they always have the *do*: label in the consequent part of the rule. This distinction facilitates the systematic search performed by the adaptation algorithm that will be discussed in the next section.

**Table 1.** POP rules partially encoding the unstack-stack strategy

<p>(1)  <b>if</b> + <math>s_0 \rightarrow</math> (on ?<math>x</math> ?<math>y</math>)  + <math>s_0 \rightarrow</math> (block ?<math>y</math>)  - <math>s \rightarrow</math> (on ?<math>x</math> table)  <b>then do</b>: <math>s'</math>: (move ?<math>x</math> ?<math>y</math> table)  <b>do</b>: <math>s_0 \rightarrow</math> (on ?<math>x</math> ?<math>y</math>) <math>s'</math></p> <p>(2)  <b>if</b> + <math>s</math>: (move ?<math>x</math> ?<math>y</math> table)  + <math>s'</math>: (move ?<math>z</math> table ?<math>w</math>)  - <math>s \rightarrow s'</math>  <b>then do</b>: <math>s \rightarrow s'</math></p>	<p>(3)  <b>if</b> + <math>s</math>: (move ?<math>x</math> ?<math>y</math> table)  + <math>s_0 \rightarrow</math> (block ?<math>y</math>)  + <math>s_0 \rightarrow</math> (on ?<math>x</math> ?<math>y</math>)  - <math>s_0 \rightarrow</math> (on ?<math>x</math> ?<math>y</math>) <math>s</math>  <b>then do</b>: <math>s_0 \rightarrow</math> (on ?<math>x</math> ?<math>y</math>) <math>s</math></p> <p>(4)  <b>if</b> + <math>s</math>: (move ?<math>x</math> table ?<math>y</math>)  - ((on ?<math>x</math> ?<math>y</math>) @ <math>s_\infty</math>)  <b>then undo</b>: <math>s</math>:(move ?<math>x</math> table ?<math>y</math>)</p>
---	---

Table 1 shows an example of POP rules in the Blocks World domain. These POP rules encode the strategy, which we call *unstack-stack*, that first unstacks all blocks to the table and then stacks them in the required configuration. This is the strategy followed to generate the plan in Fig. 4. The first POP rule unstacks block ? $x$  to the table. The first two conditions check if block ? $x$  is on top of another block ? $y$  in the initial state. The third condition checks that no existing step unstacks ? $x$  to the table. This rule makes two refinements: it adds a step  $s'$  unstacking ? $x$  to the table and adds a causal link connecting the step  $s_0$  to achieve a precondition of  $s'$ . The second POP rule ensures that unstacking steps (e.g., step  $s$ ) are done before stacking steps (e.g., step  $s'$ ). The third POP rule is intended as a refinement of an input plan so that it commits to the encoded strategy. It checks if a block (? $x$ ) that is unstacked by a step  $s$  is linked to the condition (on ? $x$  ? $y$ ) in the initial state. If it is not, it adds a causal link connecting the condition and  $s$ . This rule can be triggered in situations where in the initial state of the retrieved plan, block ? $x$  was on top of a block ? $y$  and later in that plan ? $x$  was unstacked to the table by an step  $s$ . This plan would not have been generated by the strategy encoded in Table 1. The fourth POP rule is a retraction rule. It removes any stacking step from the table that does not achieve a goal.

Any step removed by the fourth rule does not need to be added back because in the stack-unstack strategy, blocks are stacked only to achieve goals. After all these steps are removed, the four POP refinement rules of Table 1 will produce incomplete plans that can be further refined without backtracking on any of the refinements made by applying these rules. This is a highly desirable property as in some domains it might be difficult to obtain a collection of POP rules that produce a complete plan. Consequently, rules can be given for the more computationally complicated details (e.g., how to achieve the goals), leaving the rest to standard POP. Ideally, the intermediate plan produced from adaptation will be easier to complete than the initial plan. The unstack-stack strategy, partially encoded in Table 1, can be fully encoded to ensure that the resulting plans are complete. Furthermore, no backtracking will be needed during the plan adaptation process. Hence, when used in DCPOP-A, these rules will result in an omniscient plan adaptation algorithm. This will be confirmed in the experimental evaluation where no backtracking occurred in any of the plan adaptation instances. We omit presenting all the POP rules due to the lack of space.

### 4.3 Domain-Configurable Partial-Order Plan Adaptation Algorithm

Fig. 5 presents the pseudocode of the proposed plan adaptation algorithm on top of POP. It receives as input the initial state, goal state, and actions. It also receives the plan to be adapted,  $\pi_{old}$ , and the POP rules  $R$  (as described in Section 4.2). The output is a complete plan solving  $(S,G,A)$  or fail if none is found. DCPOP-A begins by adjusting  $\pi_{old}$  relative to  $(S,G)$  (Line 1). Adjust plan works by repeatedly (1) removing a step  $s$  that mentions objects in the retrieved plan that are not mapped into objects in the new problem, and (2) removing any ordering constraint or causal link connecting to/from  $s$ . This is a common step for adaptation in first-principles POP planning (e.g., [24], [13], [25]). Then, a set of plans is found by repeatedly applying retraction rules in  $R$  until none is applicable (Line 2). These plans are added to  $P$ , the list of current candidate plans to be refined. The next part of the pseudocode continues iterating while there is at least one candidate plan to be refined and no solution has been found (lines 3-14). When the list of

```

Procedure DCPOP-A( $S, G, A, \pi_{old}, R$ )
//input: initial state  $S$ , goals  $G$ , actions  $A$ , plan  $\pi_{old}$ ,
POP rules  $R$ 
//output: complete plan for  $(S,G)$  or fail
1.  $\pi_{adj} \leftarrow \text{adjust-plan}(S, G, \pi_{old})$ 
2.  $P \leftarrow \text{doAllDCRetractions}(\pi_{adj}, R)$ 
3. while ( $P \neq \emptyset$ ) do
4.    $\pi \leftarrow \text{heuristicSelectPlan}(P, A)$ 
5.    $P \leftarrow P - \{\pi\}$ 
6.   if  $\text{flaws}(\pi) = \emptyset$  then
7.     return  $\pi$ 
8.   else
9.      $P' \leftarrow \text{doOneStepDCRefinements}(\pi, R)$ 
10.    if ( $P' = \emptyset$ ) then
11.       $f \leftarrow \text{heuristicSelectFlaw}(\pi)$ 
12.       $P \leftarrow P \cup \text{refinements}(\pi, f, A)$ 
13.    else
14.       $P \leftarrow P \cup P'$ 
15. return fail

```

Fig. 5. Pseudo-code of DCPOP-A

candidate plans is empty, a failure is returned (Line 15). At each iteration, a candidate plan  $\pi$  is selected using the heuristics and is removed from  $P$  (lines 4 and 5). If this candidate plan has no flaws, it is returned (lines 6 and 7). Otherwise each plan computed by applying an applicable POP rule to  $\pi$  is added to  $P$  (lines 9 and 14). If no domain configurable refinements are found, standard POP refinements are added to  $P$  (lines 11 and 12). In principle, DCPOP-A could use any relevant plan and flaw selection heuristics described in [26] for lines 4 and 11; however our implementation uses last-in-first-out selection for both plans and flaws.

## 5 Example of Domain-Configurable Plan Adaptation

Fig. 6 shows an example of a plan obtained by adapting the plan from Fig. 4 using the unstack-stack strategy partly encoded in Table 1. The new problem has almost the same initial state as before with the exception that block F does not exist, and there are several differences in the goals. Underlined steps indicate steps retained from the retrieved plan. Continuous lines indicate causal links and ordering constraints retained from the retrieved plan (only a subset is shown). Dashed lines indicate new causal links and ordering constraints added (only a subset is shown). The steps marked “x” in Fig. 4 are steps that have been removed; The steps  $move(E,F,table)$  and  $move(F,table,E)$  were removed by adjust-plan because F does not occur in the new problem. The step  $move(C,table,A)$  was removed by the fifth POP rule because it is inconsistent with the goal. The step  $move(B,table,C)$  was added by the third POP rule because it achieves a goal. In this specific example the maximum possible number of steps is retained from the retrieved plan. In general, this is not the case because steps that could have been retained to form a complete plan will be removed if they are inconsistent with the unstack-stack strategy.

Recall from the example in Fig. 3 that Case 1 solves  $(state\ 1, final)$  and Case 2 solves  $(state\ 2, final)$ . A caveat must be made that the search space in Fig. 3 represents states of the world whereas DCPOP-A’s search space is a space of plans. However a mapping can be made from the state space to the plan space such that any transition made between states represents the corresponding action being added to the plan in the transition between plans. Continuing with the example, if we apply the unstack-stack strategy to the new problem  $(state\ 3, final)$ , then for both cases it will take the path for node p1 (meaning it will unstack all blocks). If Case 1 is being adapted then it will follow the plan laid out from p1 all the way to the goal state. If Case 2 is being adapted then it will add the step stacking C on D from p1 and then continue the rest of the plan from Case 2, which stacks the remaining blocks B and A in that order. So it takes 2 refinements if Case 1 is reused and 3 refinements if Case 3 (solves problem  $(state\ 3, final)$ ) is reused and no exploration of failed nodes is made.

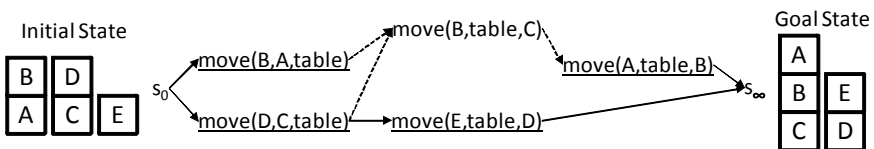


Fig. 6. Adapted partial-order plan

## 6 Empirical Evaluation

We performed experiments by encoding POP rules for the logistics transportation domain and blocks world. In the logistics transportation domain, packages must be relocated into target locations. There are two transportation means: trucks, which can be used to relocate packages within locations in the same city and airplanes, which can be used to relocate packages that are in different cities. The blocks world is a puzzle-like domain in which piles of blocks on a table must be reconfigured into a target configuration. The basic restriction is that blocks can only be moved either from the top of a pile to the top of another pile or to the table. We encoded the unstack-stack strategy described in Section 4.2 for the blocks world; therefore we attained an omniscient plan adaptation algorithm.

### 6.1 Transportation Domain Encoding

For the logistics transportation domain we encoded the following basic strategy:

1. We remove steps from the retrieved plan that load and unload any packages not at the destination city. So for example if a package  $p_5$  needs to be relocated to  $loc_2$  in  $city_3$ , then any load and unload steps of that package in any city other than  $city_3$  will be removed. This eliminates potential threats that would cause backtracking.
2. We take advantage of any steps in the plan relocating a package into the destination location in the destination city by keeping those steps and adding connecting steps as needed. So, for example, if the retrieved plan relocates  $p_5$  from a location  $loc_6$  in  $city_3$  to  $loc_2$  but in the new problem the package begins at  $airport_4$  in  $city_3$ , then steps are added that relocate  $p_5$  into  $loc_6$  from  $airport_4$ .
3. We added steps to the plan that relocate packages to the destination city if needed, taking advantage of any existing steps driving a truck or flying an airplane whenever possible. So for example, if  $p_5$  was initially in  $loc_7$  in  $city_2$ , then it will be relocated to an airport in  $city_2$ . If an existing step in the plan already moves  $p_5$  from  $loc_7$  to an airport in  $city_2$ , this step will be reused. Otherwise a new step will be created. Steps are also added relocating  $p_5$  from  $airport_1$  to  $airport_4$ . If an existing step in the plan flies  $p_5$  from  $airport_1$  to  $airport_4$  it will be reused. Otherwise a new step is created.

These encodings ensure that DCPOP-A will be omniscient when solving problems in the logistics transportation domain.

### 6.2 Experimental Setup

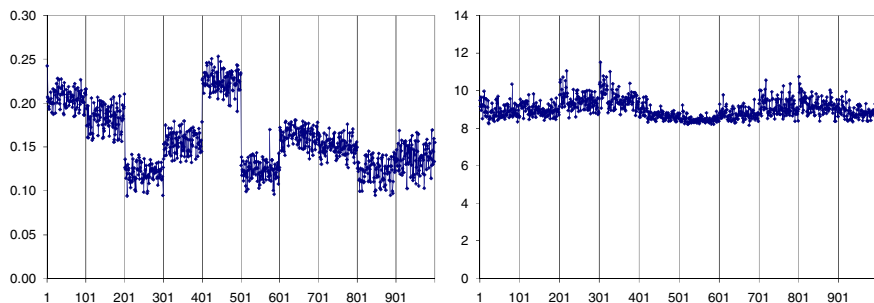
For each domain we constructed a case base of 100 cases and a testing set of 10 problems. All problems have the same goals but their initial state is randomly generated. For the blocks world the goal is to achieve a 5-block pile and for logistics a particular configuration of 4 packages required to be at 4 different locations. The initial state for the blocks world is a configuration of the 5 blocks. So the total number of problems that can be generated is 501. The initial state for the transportation domain is a configuration of 3 cities, each having 3 locations (including 1 airport), each city has 1

truck and there are 2 airplanes. So the total number of problems that can be generated, given that the packages can start in any of the 9 locations and that the start locations of the trucks and airplanes are fixed, is 6561. For each problem  $p$  in the testing set we adapt each of the cases  $c$  stored in the case base. We run each problem-case pair  $(p, c)$  30 times and average the results. So the total runs for each domain was  $10 * 100 * 30 = 30,000$  runs.

Fixing the goals is a simplifying way to simulate how our retrieval algorithms would work with an omniscient adaptation algorithm. Namely, we will simply retrieve any case that achieves the same goals regardless of the similarity. In experiments reported in [27], it is shown that modifying features in the initial state can result in a significant change in the adaptation process on top of a partial order planner. For historical context, in Prodigy/Analogy [1] retrieval occurs by iterating two steps. At the first step the system uses a hash table to identify if there are cases stored achieving the same number goals and then, in the second step, computes similarity based on the initial state. If a sufficiently similar case is found (e.g., the similarity of the initial states is greater than a pre-defined threshold) then the case is retrieved. Otherwise it repeats the two steps by removing one goal. With an omniscient adaptation algorithm the second step would be unnecessary. A similar process to Prodigy/Analogy is performed in CAPlan/CbC and derSNLP.

### 6.3 Results

Fig. 7 shows the run-time results for the blocks world (left) and the logistics transportation domain (right) respectively. The x axis corresponds to the 100 cases \* 10 problems and the y axis correspond to the average time in seconds over the 30 runs for each (case, problem) adaptation process. The x-axis is sorted so the first 100 averaged data points are shown with the first given problem, then again the next 100 points with the second given problem, and so forth. Thus, the vertical bars in the graphs separate data for each of the 10 problems; between those bars (i.e., for a given problem), the data points show the averaged times to adapt each of the cases in the CB into a solution for the problem. In the blocks world domain, we observe that the running times for adapting each case to a given problem is clustered around the same time intervals. For example, for the 4<sup>th</sup> problem the average time to adapt all cases is



**Fig. 7.** Adaptation times for blocks world (left) and logistics (right)

0.155 seconds with a standard deviation of 0.012 seconds. We observed similar results across all other problems. In the logistics domain, there is no significant time difference between solving times across all given problems; the average problem solving time, across all pairs (case, problem), is 9 seconds with a standard deviation of 0.5 seconds. The results for both domains support our hypothesis that regardless of which any two cases are retrieved for a given problem, their adaptation times will be roughly the same, regardless of the individual cases similarity to the new problem, and given that the problems solved by the cases and the new problem are of the same size.

## 7 Conclusions

In this paper we revisited the adaptation and retrieval trade-off traditionally held as a principle in case-based reasoning research and even practice. We argue that this principle involves an implicit assumption that the adaptation algorithm is capable of solving problems fast enough only when the retrieved case is reasonably similar to the input problem. If the retrieved case is not sufficiently similar to the new problem then the adaptation effort will take a significant amount of time. To make this assumption explicit, we analyzed this principle under two boundary conditions: (1) for naïve and (2) for omniscient adaptation algorithms. Using a simple complexity analysis, we conclude the adaptation-retrieval trade-off does not necessarily exist for the naïve adaptation procedures. We also claim that the adaptation-retrieval trade-off does not necessarily hold for connected domains, and validated this hypothesis empirically on two classical connected domains used widely in the case-based planning literature.

A provocative implication of our results is that, because for omniscient plan adaptation there is no adaptation-retrieval trade-off, we can substantially reduce the case base by simply having one case for each combination of goals indexed by a hash table so that the retrieval procedure would run in constant time by simply identifying the case that achieves the same goals (or even use an empty CB and plan from scratch). However, running time is but one criterion by which we can measure the effectiveness of the overall case-based reasoning process. Other, arguably at least as important criteria, such as quality of the resulting plan, should be considered. Indeed, one of the motivational scenarios for case-based reasoning is a case base storing a collection of hand-crafted solutions. In this scenario, the retrieval task is to find a very similar case, if not the most similar one, and the adaptation task is to commit to the retrieved plan as much as possible. Throwing away half of the solution, as encoded in the unstack-stack strategy, would not make any sense in this scenario regardless of how fast the adapted solution is generated. Instead we envision highly tuned POP rules that are sensible towards retaining crucial steps identified by the user and retrieval procedures that ensure that plans meeting certain constraints are produced, as recent research on retrieval has suggested (e.g., [28], [29]).

For future work we are planning to investigate the adaptation-retrieval trade-off for non connected domains such as the logistics transportation domain with one-way routes. Unlike connected domains, there is no guarantee that the adjusted plan can always be extended to reach a solution. Hence, the question is whether POP rules can be written that rapidly remove parts of the adjusted plan in such a way that (1) a significant portion of the retrieved plan is reused in the adjusted plan and (2) this adjusted plan can rapidly be refined to obtain a solution.



**Acknowledgements.** This research was supported by grants from the Air Force Research Laboratory and the National Science Foundation Grant No. NSF 0642882.

## References

1. Veloso, M.M.: Planning and Learning by Analogical Reasoning. Springer, Heidelberg (1994)
2. Muñoz-Avila, H., Weberskirch, F.: Planning for Manufacturing Workpieces by Storing, Indexing and Replaying Planning Decisions. In: AIPS 1996, pp. 150–157. AAAI Press, Menlo Park (1996)
3. Ihrig, L., Kambhampati, S.: Storing and Indexing Plan Derivations through Explanation-Based Analysis of Retrieval Failures. JAIR 7, 161–198 (1997)
4. Koehler, J.: Avoiding Pitfalls in Case-based Planning. In: AIPS 1994, pp. 104–109 (1994)
5. Francis, A., Ram, S.: A Comparative Utility Analysis of Case-based Reasoning and Control-rule Learning Systems. In: Lavrač, N., Wrobel, S. (eds.) ECML 1995. LNCS, vol. 912. Springer, Heidelberg (1995)
6. Gomes, P., Pereira, F.C., Seco, N., Pavia, P., Carreiro, P., Ferreira, J., Bento, C.: Combining Case-Based Reasoning and Analogical Reasoning in Software Design. In: Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science (2002)
7. Smyth, B., McKenna, E.: Footprint-Based Retrieval. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS, vol. 1650, p. 343. Springer, Heidelberg (1999)
8. Chang, H., Dong, L., Liu, F., Lu, W.: Indexing and Retrieval in Machining Process Planning Using Case-Based Reasoning. Artificial Intelligence in Engineering 14 (2000)
9. Bonzano, A., Cunningham, P., Smyth, B.: Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 291–302. Springer, Heidelberg (1997)
10. Stahl, A., Gabel, T.: Using Evolution Programs to Learn Local Similarity Measures. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689. Springer, Heidelberg (2003)
11. Ihrig, L., Kambhampati, S.: Design and Implementation of a Replay Framework Based on a Partial Order Planner. In: AAAI/IAAI 1996, pp. 849–854. AAAI Press, Menlo Park (1996)
12. Muñoz-Avila, H., Weberskirch, F.: A Case Study on the Mergeability of Cases with a Partial-Order Planner. In: Steel, S. (ed.) ECP 1997. LNCS, vol. 1348, pp. 325–337. Springer, Heidelberg (1997)
13. van der Krogt, R., de Weerd, M.: Plan Repair as an Extension of Planning. In: ICAPS 2005, pp. 161–170. AAAI Press, Menlo Park (2005)
14. Gerevini, A., Serina, I.: Fast Plan Adaptation through Planning Graphs: Local and Systematic Search Techniques. In: Proc. of AIPS 2000, pp. 112–121. AAAI Press, Menlo Park (2000)
15. Warfield, I., Hogg, C., Lee-Urban, S., Muñoz-Avila, H.: Adaptation of Hierarchical Task Network Plans. In: FLAIRS 2007, pp. 429–434 (2007)
16. Goel, A., Ali, K., Donnellan, M., Gomez, A., Callantine, T.: Multistrategy Adaptive Navigational Path Planning. IEEE Expert 9(6), 57–65 (1994)
17. Tonidandel, F., Rillo, M.: Case Adaptation by Segment Replanning for Case-Based Planning Systems. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS, vol. 3620, pp. 579–594. Springer, Heidelberg (2005)

18. Fikes, R., Nilsson, N.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2, 189–208 (1971)
19. Nau, D., Au, T., Ilghami, O., Kuter, U., Muñoz-Avila, H., Murdock, J., Wu, D., Yaman, F.: Applications of SHOP and SHOP2. *IEEE Intelligent Systems* 20(2), 34–41 (2005)
20. Knoblock, C.: Generating Parallel Execution Plans with a Partial-Order Planner. In: *AIPS 1994*, pp. 98–103. AAAI Press, Menlo Park (1994)
21. Paulokat, J., Wess, S.: Planning for Machining Workpieces with a Partial-Order, Nonlinear Planner. In: *Proceedings of the AAAI 1994 Fall Symposium on Planning and Learning*, AAAI Press, Menlo Park (1994)
22. Nguyen, X., Kambhampati, S.: Reviving Partial Order Planning. In: *IJCAI 2001*, pp. 459–466. Morgan Kaufmann, San Francisco (2001)
23. Vidal, V., Geffner, H.: Branching and Pruning: An Optimal Temporal POCL Planner Based on Constraint Programming. *Artificial Intelligence* 170(3), 298–335 (2006)
24. Hanks, S., Weld, D.S.: A Domain-Independent Algorithm for Plan Adaptation. *JAIR* 2, 319–360 (1995)
25. Kuchibatla, V., Muñoz-Avila, H.: An Analysis on Transformational Analogy: General Framework and Complexity. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 458–473. Springer, Heidelberg (2006)
26. Younes, H., Simmons, R.: VHPOP: Versatile Heuristic Partial Order Planner. *Journal of Artificial Intelligence Research* 20, 405–430 (2003)
27. Muñoz-Avila, H., Hüllen, J.: Feature Weighting by Explaining Case-Based Planning Episodes. In: Smith, I., Faltings, B.V. (eds.) *EWCBR 1996*. LNCS, vol. 1168, pp. 280–294. Springer, Heidelberg (1996)
28. McSherry, D.: Completeness Criteria for Retrieval in Recommender Systems. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 9–29. Springer, Heidelberg (2006)
29. Nicholson, R., Bridge, D., Wilson, N.: Decision Diagrams: Fast and Flexible Support for Case Retrieval and Recommendation. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 136–150. Springer, Heidelberg (2006)

# Boosting CBR Agents with Genetic Algorithms

Beatriz López, Carles Pous, Albert Pla, and Pablo Gay

University of Girona,  
Campus Montilivi, edifice P4, Girona, Spain  
{beatriz.lopez, carles.pous}@udg.edu,  
{apla, pgay}@eia.udg.edu  
<http://exit.udg.edu>

**Abstract.** In this paper we present a distributed system in which several case-based reasoning (CBR) agents cooperate under a boosting schema. Each CBR agent knows part of the cases (a subset of the available attributes) and is trained with a subset of the available cases (so not all the agents know the same cases). The solution of the system is then computed by means of a weighted average of the solutions provided by the CBR agents. Weights are actively learnt by a genetic algorithm. The system has been applied to a breast cancer application domain. The results show that with our methodology we can improve the results obtained with a case base in which attributes have been manually selected by physicians, saving physicians work in future.

**Keywords:** Distributed CBR, genetic algorithms, boosting, multi-agent systems.

## 1 Introduction

Distributed environments offer a new way of addressing case-based reasoning (CBR) approaches [24]. For example, we can design a multi-agent platform in which several agents cooperate in the solution to new problems with different experiences (or case bases); we can have agents with different domain models, so that each one can be specialized in a particular field of knowledge; or we can have agents with different competence models, thus they follow a CBR approach with different methods and criteria. From a machine learning point of view, this distributed approach of CBR based agents can be considered as ensemble learning [21].

Ensemble learning with CBR agents has been introduced in recent years [17][15]. Among the principal kinds of ensemble techniques, we have boosting, bagging and stacking. In a boosting scenario, a new problem is solved as the weighted vote of all the agents' solutions. Each agent has its own case base, that has been actively designed to coordinate agents' experiences. Second, bagging consist of using a weighted voting schema, but in this case experiences are randomly assigned to each agent so the construction of complementary learners is left to chance and to the variability of the learning methods [27]. Finally, in staking techniques each agent follows different reasoning techniques. So, we can

have CBR agents with different similarity measures or decision criteria. Several authors have also studied how experiences should be stored in the overall system [16].

Our work is related to boosting CBR agents since we are actively seeking complementarity of learners. Therefore, the key issue is the weight assigned to each agent [1]. AdaBoost [8] is one of the best known learning algorithms for this purpose, which today has a lot of variants. However, Adaboost exhibits greedy behavior and recent papers [28] argue in favor of other approaches, such as genetic algorithms, that outperform Adaboost. Our research is related to extend the application of genetic algorithms, for boosting purposes, in a multi-agent environment where CBR agents cooperate to solve a problem. Particularly, our CBR agents only know part of the cases, as in [15], following the implicit knowledge distribution of an organization. Thus, each agent solves a problem by means of CBR, and the solution is then combined by a coordinator agent who has learnt the agent's weight according to a GA.

This paper is organized as follows. First we introduce the boosting schema in which our CBR agents cooperate. Next, we describe the GA we propose. We continue by providing the information about the application domain we are working on and the results obtained. Finally, some related work is highlighted and some conclusions and discussion are provided.

## 2 Boosting CBR Agents

Our boosting approach consists of  $n$  case-based agents that cooperate for solving a problem as in a multi-agent system (MAS). Each agent provides its advice or solution about a case, and a coordinator makes a final decision based on a weighted voted schema (see Figure 1).

Each agent is trained with a set of examples that can differ from one agent to the other. Each agent receives only a part of the examples' attributes (as in [15]). Thus, each agent is specialized in a particular field of knowledge of the domain. That is, if cases are composed of attribute-value pairs  $\langle (a_1, v_1), \dots, (a_m, v_m) \rangle$ , then each agent  $j$  has information related to a subset of these attributes, that we represent as follows:

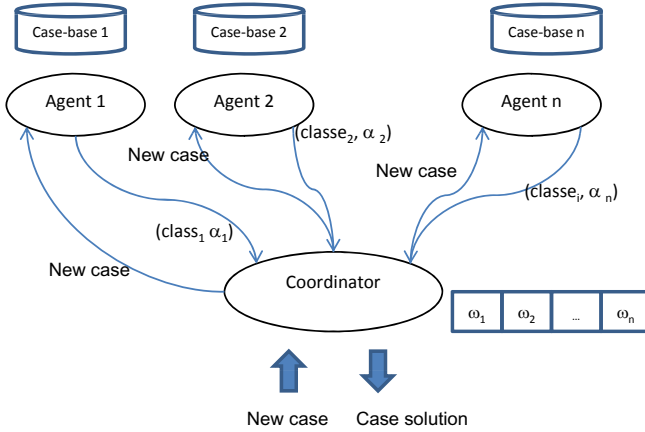
$$\langle (a_1^j, v_1^j), \dots, (a_{n_j}^j, v_{n_j}^j) \rangle \quad (1)$$

The attribute sets among the different agents are disjoint, so:

$$\{a_1^j, \dots, a_{n_j}^j\} \cap \{a_1^k, \dots, a_{n_k}^k\} = \emptyset \quad (2)$$

When a new case  $C$  needs a solution, the coordinator agent broadcasts the case to the CBR agents. CBR agents compute internally a solution for the case following

<sup>1</sup> This is a particularly view of boosting, since, traditionally, boosting is sequentially adding learners when the previous fails in some instances, while we start with a fixed set of classifiers (with different instance sets) and we are looking the complementarity by their weights.



**Fig. 1.** Agent's interaction

a case-based reasoning process. That is, in the retrieval phase of the CBR cycle, each agent,  $j$ , evaluates the similarity between the new case  $C$  and each case  $C_i^j$  in its memory as follows:

$$Sim_j(C_i^j, C) = \frac{\sum_{x=1}^{n_j} w_x^j * simAt(v_x^j, v_x)}{\sum_{x=1}^{n_j} w_x^j} \quad (3)$$

where  $w_x^j$  is the internal weight used for the  $j$  agent regarding the  $x$  attribute;  $n_j$  the number of attributes handled by the  $j$  agent; and  $simAt(v_x^j, v_x)$  the similarity function at the attribute level between the  $j$  agent's case memory attribute  $a_x^j$  and the new case attribute  $a_x$ . Note then, that each agent has its own similarity measure  $Sim_j(X, Y)$  and knowledge related to the weight it assigns to each attribute. All cases retrieved with a similarity higher than a given threshold  $\lambda$  are selected and passed to the reuse phase.

Each agent follows the same reuse method in order to determine the solution of a case given the selected cases  $C_{i_1}^j, \dots, C_{i_k}^j$ . Particularly, we are considering diagnosis domains with only two possible solutions: positive class (or illness) and negative class (or healthy). Thus, we define a decision variable  $\delta_j$  that relates the number of positive solution in the selected cases rated to the total number of cases selected [19], as follows:

$$\delta_j = \frac{\sum_{C_i^{j,+}} Sim_j(C_i^{j,+}, C)}{\sum_{x=1}^k Sim_j(C_{i_x}^j, C)} \quad (4)$$

**Table 1.** Example of the individual agent’s answers

	class	confidence	weight
		$\alpha_j$	$w_j$
agent 1	+	0.7	0.8
agent 2	+	0.6	0.2
agent 3	-	0.9	0.3
agent 4	+	0.8	0.1
agent 5	-	0.7	0.7
agent 6	-	0.9	0.3
agent 7	+	0.8	0.2
agent 8	-	0.9	0.4

where  $C_i^{j,+}$  are the selected cases with positive class as a solution, and  $C_{i_x}^j$  the  $x$  case selected by the  $j$  agent. Whenever we get enough evidence about the positive class, the solution is +. That is, if  $\delta_j > \tau$ , where  $\tau$  is a given threshold, then the CBR agent’s answer is the positive class.

Next, the CBR agents reply to the coordinator with a tuple  $\langle class_j, \alpha_j \rangle$  that contains the class to which the case belongs according to its case-base, say  $class_j$ , and the confidence  $\alpha_j$  it has on that solution, a value in  $[0,1]$ , where 1 means highly confident.

The confidence value is obtained through the  $\delta_j$  value, and taking into account the results of the reuse phase (equation 4), and so it is related to the work of Cheetham 4. Hence, the  $\alpha_j$  value is set according to the following rule:  $\alpha_j = \delta_j$  if  $class_j = +$  (as in equation 4 we look for the positive evidence), or  $\alpha_j = 1 - \delta_j$  if  $class_j = -$ .

Afterwards, the coordinator agent combines the different answers in order to find information related to the final solution. For this purpose, the coordinator has a weight  $w_i$  on each agent concerning the reliability on the information provided by the agents (as trust values 2). Thus, the coordination gathers all the evidence regarding the positive class ( $v^+$ ) and all the evidence regarding the negative class ( $v^-$ ), as follows:

$$v^+ = \frac{\sum_{class_j=+} \omega_j * \alpha_j}{\sum_{class_j=+} \omega_j} \qquad v^- = \frac{\sum_{class_j=-} \omega_j * \alpha_j}{\sum_{class_j=-} \omega_j} \qquad (5)$$

Note, then, that we are computing the average of the results of the individual CBR agents, regarding positive and negative classes. If  $v^+ \geq v^-$ , then the final answer of the multi-agent system is the positive class with the  $v^+$  evidence value. Otherwise, it is the negative class with the  $v^-$  evidence value. 2

For example, let us suppose that we have a boosting system composed of eight CBR agents, and that they reply to the coordinator agent with the outcomes

<sup>2</sup> Note, that our approach based on positive and negative classes is performed before obtaining any feedback; so it differs from the measures proposed in 6.

shown in Table 1. Let also suppose that the coordinator has the weight values for each agent expressed in the third column of the table. Then, the following evidences for the positive and negative classes are computed:  $v^+ = 0.71$  and  $v^- = 0.82$ . Thus, the solution of the case is "negative", with an evidence of 0.82.

### 3 Learning Boosting Weights with GA

In the previous section we have seen how the CBR agents are organized in a boosting schema in order to obtain a solution to a problem. The procedure depends on the weight each agent has in the system (see equation 5). In this section we explain how these weights are obtained via a genetic algorithm.

Our genetic algorithm (GA) consists of the following steps, as any classical GA [13]:

1. Start with a randomly generated population of chromosomes
2. Calculate the fitness of each chromosome in the population
3. Repeat
  - (a) Select a pair of chromosomes from the current population
  - (b) With a probability  $p_c$  crossover the pair to form two offspring
  - (c) With a probability  $p_m$  mutate the two offspring
4. Replace the current population with the new one (reinsertion)
5. Goto step 2 until some ending condition holds.

As it is possible to observe in the algorithm, randomness is involved in this kind of algorithms and several runs are required in order to deal with it [13]. Hence, two runs with different random-number seeds will generally produce different detailed behaviours. But finally, a single, averaged weight should be assigned to a boosting agent. Thus, we assume that we have a set cases (or examples),  $BC$ , from which the weights can be learnt. Then, by using a cross-validation methodology, we define  $k$ -folds,  $fold^1, \dots, fold^k$ , each one containing a set of training cases for building a case base,  $bc_i$ , and a set of test cases,  $test_i$ .

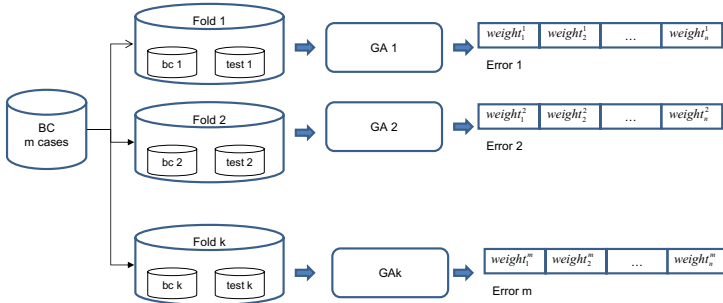


Fig. 2. GA runs

**Table 2.** GA for weight learning. Observe that we are using array operations

```

Let be  $W$  the array of weights
Let be  $BC$  the set of cases
Define  $k$  folds,  $f^i \subset BC$ 
For each fold  $i=1, \dots, k$ 
     $W^i \leftarrow$  Run GA
end for
 $W = \frac{\sum_{i=1}^k W^i}{k}$ 

```

We apply the GA once on each fold. From each fold, we obtain a weight vector  $W^i$  (one weight for each agent, see Figure 2), and finally, we set the agents' weights with the averaged results  $W$ . See in Table 2 the corresponding algorithm.

Next, we detail how the chromosome is defined, the fitness function, the selection, crossover and mutation operation, the reinsertion technique and the ending condition.

### 3.1 Chromosome

Each chromosome  $cr_x$  in the population represents a weight assignment in the multi-agent system. Thus, the chromosome is an array of real variables in  $[0,1]$  of  $n$  length, where  $n$  is the number of CBR agents in the system. The real variable  $i$  in the array represents the weight assigned to the agent  $i$  in the boosting schema.

### 3.2 Fitness Function

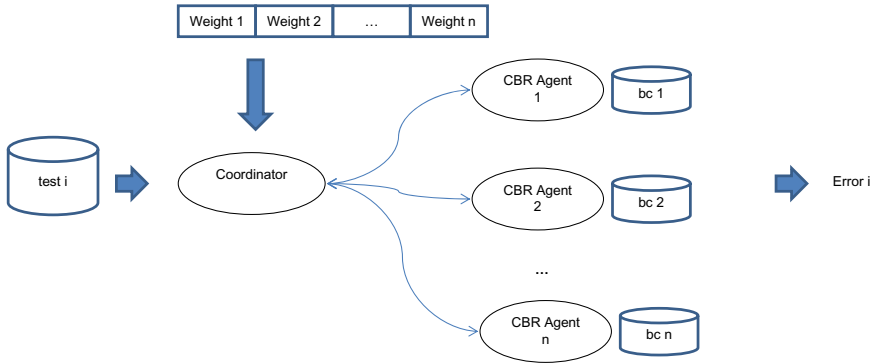
We want to minimise the error of the boosting CBR system according to a given case base. Thus, in order to obtain the fitness value of a chromosome, we built the boosting CBR system it represents, we use the training set of the current fold  $i$  as the case base, we run the test cases, and we obtain an error value (see Figure 3).

Particularly, we can define the error of the boosting CBR system codified in a chromosome  $cr_x$  as the average of the individual errors for each case in  $test_i$ . Formally,

$$error(c_x) = \frac{\sum_{j=1}^l |real_j - predicted_j|}{l} \quad (6)$$

where  $l$  is the number of cases with which the GA is tested;  $real_j$  is the real class or solution of the  $j$  test case (1 for the positive class and 0 for the negative class); and  $predict_j$  is the evidence predicted by the boosting CBR system. That is,  $predict_j = v^+$  if the class is +, and  $predict_j = 1 - v^-$  if the class is negative.





**Fig. 3.** Decodification process of a chromosome into a boosting CBR system to test it and obtain the error rate

Observe, that in the optimal case in which the results of the boosting CBR system was perfect, then the error is 0; while in the worse case (an error of 1 for each test case), the error is 1.

Since GA maximize the fitness function, and we need to minimize the error, we define the fitness in our GA as the inverse of the error as follows,

$$fitness(cr_x) = (1 - error(cr_x))^3 \tag{7}$$

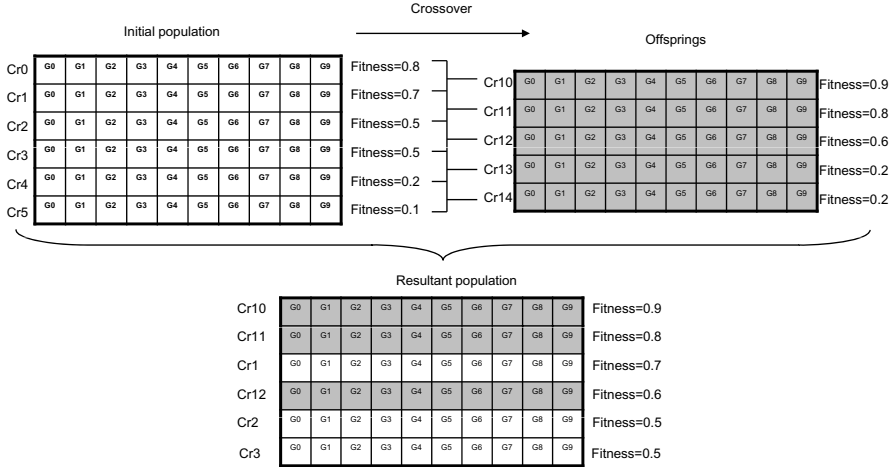
### 3.3 Selection

Selection determines which individuals are chosen for mating [18]. We have followed a truncation selection schema: individuals are sorted according to their fitness and only the best individuals are selected for parents. Particularly, we have chosen a proportion of 50% as truncation threshold. That means that the 50% of the best parents have been selected. This value has been studied in [3], and defines a selection intensity of 0.8; that is, the loss of diversity and variance of the individuals is close to the mean possible value.

Other alternatives, such as roulette-wheel selection [13] have been tried too, but the truncation selection has produced better results.

### 3.4 Crossover

Crossover define how the offspring are generated from the parents. We are using the uniform crossover [14]: a crossover bit mask is generated at random with the same length as the chromosome structure and the values in the mask indicate which parent will supply the variable value of the offspring. If there is a 1 value, the variable value is taken from the first parent; otherwise, the variable value is taken from the second parent. This method is identical to discrete recombination [18].



**Fig. 4.** Elitism schema for reinsertion

Particularly, we define a parameter  $\eta$  that we initialize to 0.5. Then, we generate a random value  $r$  in  $[0,1]$ . If  $r \leq \eta$ , the bit mask is set to 1; otherwise to 0.

### 3.5 Mutation

After recombination offspring can be mutated with low probability [13]. Thus, a random value  $\varphi$  is generated; if  $\varphi > p_m$ , where  $p_m$  is the mutation probability (see the mutation step of the GA), then the offspring is mutated.

The mutation procedure is the following:

1. Let  $p_m$  be the mutation probability
2. For each variable  $y$  of the chromosome  $cr_x$ 
  - (a)  $r_m =$  random value in  $[0,1]$
  - (b) If  $r_m \geq p_m$  then  $cr_x(y) =$  random value in  $[0,1]$ .

### 3.6 Reinsertion

After producing offspring a new population should be determined [14]. We are using an elitism schema: at each generation, the best offspring and parents replace the population. Elitism schemas are maybe the most recommended method since they prevent loss of information in case that the parents do not produce better offspring. In such a situation, the average fitness of the population can decrease. Thus, using an elitism schema, the best individuals can live for many generations.

Figure 4 graphically shows how the elitism schema works in our GA. On the left, we have the initial population with six chromosomes,  $cr_0, \dots, cr_5$  each of

them representing different boosting CBR systems (through the different weights they codify). They are sorted according to their fitness, being the top one the chromosome with the highest fitness value. After crossover, we obtain the offspring on the right. There are some offspring with a higher fitness than their parents, but there are also some others with a worse fitness. Then, the reinsertion operator takes the best of all of them, resulting in the new chromosomes (next population or generation) shown at the bottom of the figure.

### 3.7 Ending Condition

We are computing the average error for all the chromosomes in the current generation  $E_i$ . Thus, when the error difference of two consecutive iterations is smaller than a given parameter  $\epsilon$  (i.e.  $|E_i - E_{i+1}| < \epsilon$ ), we run the GA for 50 additional iterations, and then we terminate it. We have set  $\epsilon = 0.05$ .

## 4 Application to Breast Cancer

In this section we provide a brief description of the case base used for experimentation. Afterwards, some details about the experimental setup are presented. Next, the results obtained are shown.

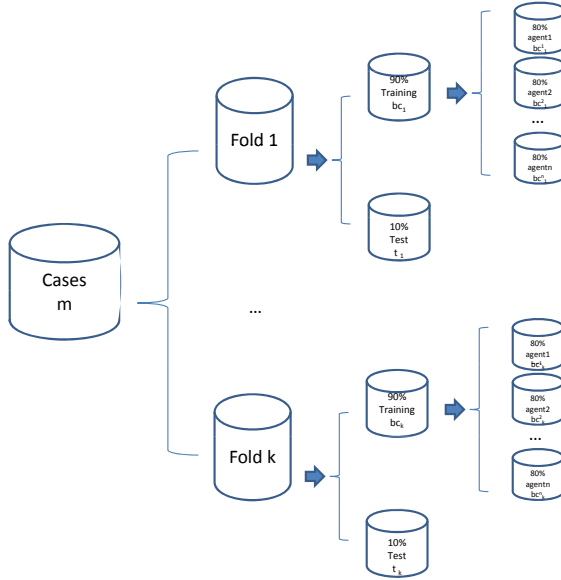
### 4.1 Breast Cancer Case Base

We have used a breast cancer data base provided by the team of physicians we are working with. It consists of 871 cases, with 628 corresponding to healthy people and 243 to women with breast cancer. There are 1199 attributes for each case. Since there are redundant, wrong and useless information a preprocess was carried out. Also, the preprocess was used to obtain data corresponding to independent individuals, since there are patients in the database that are relatives. As a consequence, the database was reduced to 612 independent cases, having 1106 attributes per case.

Data has been partitioned in 8 groups, following the questionnaire criteria with which physicians have collected the data and that are related to different medical specialisations (epidemiology, family information, etc.). Each group of data has been assigned to an agent, resulting in 8 CBR agents in our system.

In addition to that, two different case bases have been considered for experimentation purposes:

- Total: in which each case is described as original provided by the physicians, that is, with the complete set of attributes (1106).
- Manual: in which the physicians have manually selected relevant attributes (85 attributes).



**Fig. 5.** Cross validation extended for boosting CBR agents

## 4.2 Experimental Set Up

The implementation of our method has been done as an extension of the eXiT\* CBR tool [20]. This tool allows us to easily define CBR agents in a parameter-based way. Then, we have extended the platform for the boosting and genetic algorithm explained in this paper.

We have followed a cross-validation procedure taking into account the boosting purposes of our method. Hence, we have defined 10 folds, with 90% of the cases for training and 10% for testing. However, regarding the training cases, different examples have been assigned to each agent (see Figure 5). Thus, among 90% of training cases, only 80% of them are assigned randomly to an agent. This assures us that there are some differences in the composition of the case bases of each agent. Future experiments should contemplate other proportions.

Therefore, with the 10 folds, 10 GA runs have been performed, one per fold. Since we have 8 agents, we finally obtain 10\*8 weights, that we averaged in the final 8 weights, one per agent. Concerning the other parameters of the GA, we have chosen the following values:

- Number of chromosomes (individuals) in the population: 50
- Chromosome length = 8, since we have  $n = 8$  agents.
- Crossover probability: always 1, since we are applying the elitism mechanism in the reinsertion procedure, it makes sense to always generate new individuals.
- Mutation probability: randomly generated for each GA run.

Our hypothesis is that we can obtain a better performance with our boosting CBR agent scenario, in which the weights are learnt via GA, than without learning weights, even if the features are selected by the physicians. Therefore, the following experimental settings have been defined:

- *TotalTrust1*: all the CBR agents weights have been set to 1, and the complete set of attributes has been used (1106).
- *TotalGA*: the CBR agents weights have been set according to the results of the GA runs, and the attributes as in *TotalTrust1*.
- *ManualTrust1*: all the CBR agents weights (as in equation 4) have been set to 1, and the attributes used to represent the cases are the ones manually selected by the physicians (85).
- *ManualGA*: the CBR agents weights have been set according to the results of the GA runs, and the attributes as in *ManualTrust1*.

The attribute weights (in equation 3) have been set to 1 in all of the experiments.

The results obtained in each experimental configuration are detailed in the next section.

### 4.3 Results

We have used ROC (*Receiver Operator Characteristics*) curves to analyse the results. ROC curves depict the tradeoff between true positives and false positives regarding to a changing threshold 7 ( $\tau$  in our case). Physicians use to work with this kind of plots and they feel comfortable interpreting the results on this graphic manner.

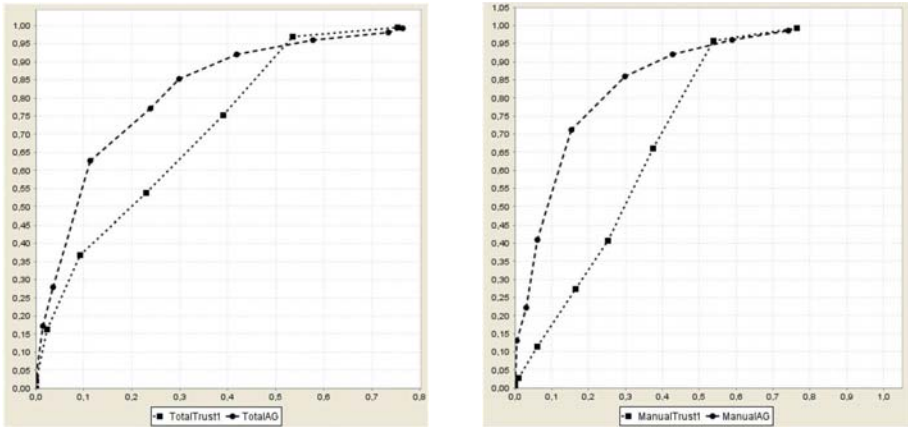
In Figure 6 left we show the results obtained in the scenarios in which all the attributes have been used. We can see how the *TotalGA* line outperforms the *TotalTrust1* line, meaning that we have significantly increased the performance of the system when learning the agents weights. This improvement can also be achieved in the scenarios in which the attributes were manually selected by the physicians, as shown on Figure 6 right.

Analysing in detail the results, we obtain the following AUC (Area Under the Curve) values:

- *TotalTrust1*: 0.770
- *TotalGA*: 0.849
- *ManualTrust1*: 0.706
- *ManualGA*: 0.853.

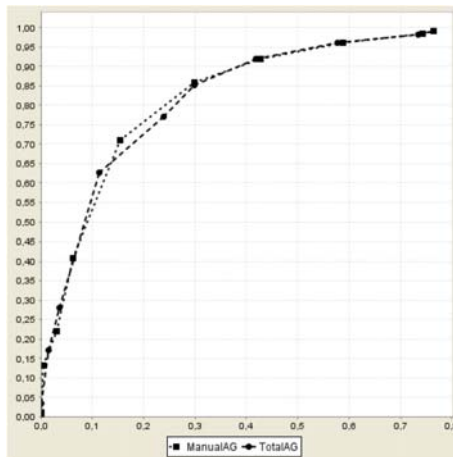
Therefore, our hypothesis is validated. What is important, here, is to remark that even for the case in which the space of attributes has been manually reduced, the results are maintained. So, the effort that the physician has dedicated to perform this selection can be saved in future. Results with our methodology do not significantly change due to the space reduction, as shown in Figure 7.

Finally, we have compared our boosting approach with a non-boosting approach. The results are quite closer, even that the non-boosting approach performs slightly better (the AUC values are 0.84 and 0.89 correspondingly).



**Fig. 6.** *Left:* Results obtained when using the complete set of attributes: *TotalTrust1* when all the agents have the same weight ( $=1$ ); *TotalGA* when using the weights learnt with our GA. *Right:* Results obtained when using the set of attributes selected by the physicians: *ManualTrust1* when all the agents have the same weight ( $=1$ ); *ManualGA* when using the weights learnt with our GA. The x-axis shows the false positive rate, while the y-axis the true positive rate.

However, we need to assume that it is not always possible to follow a centralized approach. As in our case, each agent corresponds to the information gathered in a particular unit of a hospital, and the information is kept in private inside each unit. So a centralized approach is not suitable.



**Fig. 7.** Comparison of the results obtained with our methodology in the two case bases: *Total*, the original one; *Manual* with the number of attributed reduced

## 5 Related Work

There are several works related to boosting CBR agents in particular, and ensemble learning in general [25][24][12]. For example, in [17] two schemas are proposed: bagging and boosting. In this work, the authors focus on how cases should be shared among agents. We are not so worried about that, but in how to set up the weights assigned to the agents for the boosting voting schema.

A work closer to ours is [15], in which the authors propose a corporate memory for agents, so that each agent knows about a piece of the case, as in our case. In [15], however, the authors propose a negotiated retrieval method based on distributed constraint optimisation techniques. We are using a simpler, but easier way of computing the solution based on weighting the vote of the CBR agents. Other similar approaches that also specialize the classifiers on particular parts of the domain problems are [26] and [5]. However, these approaches focus on learning different attribute weights, instead of the voting weight of the classifier.

Regarding research works on the use of GA in a boosting environment, it is important to distinguish the approach followed in [28]. Here, the authors analyse the greedy behaviour of Adaboost and suggest to use GAs to improve the results. Another interesting work is [9], in which the GAs are used to search on the boosting space for sub-ensembles of learners. However, none of these approaches are using CBR or distributed CBR approaches as we are doing. In [22] a GA is also used to combine several k-nearest neighbor approaches in a more similar to our approach, in the sense the authors are using GA to select the appropriate classifier.

Other applications of GA in CBR are related to the retrieval phase for feature and instance learning, as in [1] and [10]. We understand that in a future we need to complement our approach with these works and study the synergies between feature learning and classifier weight learning, together with local similarity measures as the ones studied in [23].

Finally, an interesting study to which our research work is related is [11]. In this case, agent's trust is studied under the evolutionary paradigm. We believe that our approach based on specialised agents is equivalent to it. This view of ensemble weights as trust has also been studied in [2].

## 6 Conclusions

Distributed case-based reasoning has been a matter of study in the recent years. We can find boosting or bagging agents that, with different case bases, cooperate in the solution of a problem. In this paper we have presented a boosting schema in which several CBR agents cooperate to solve a new problem. Our agents do not know the complete case, but a piece of it. In addition, each CBR agent has a different case-base. Weights related to the voting schema implicit in the boosting organization are learnt by means of a GA.

We have described how these weights can be coded in a chromosome and all the parameters involved in a GA in order to evolve these candidate weights to some other ones that improve the boosting system.

We have applied our method to develop a boosting CBR system to deal with breast cancer diagnosis. We have obtained successful results, even when the number of attributes varies.

As future work, we need to explore other ways to combine our CBR agents. For example, we should consider different methods and criteria in order to fit the most suitable CBR technique to the available data. That is, until now, we have defined the set of agents according to different domain specializations. So, each agent knows about a subset of the different attributes related to a case. Other issues, such as the criteria or methods to deal with the attributes should be studied in future.

**Acknowledgments.** This research project has been partially funded by the Spanish MEC projects DPI 2006-09370, CTQ2008-06865-C02-02/PPQ, TIN2008-04547/TIN, and Girona Biomedical Research Institute (IdiBGi) project GRCT41.

## References

1. Ahn, H., Kim, K.-j., Han, I.: Hybrid genetic algorithms and case-based reasoning systems. In: Zhang, J., He, J.-H., Fu, Y. (eds.) CIS 2004. LNCS, vol. 3314, pp. 922–927. Springer, Heidelberg (2004)
2. Birk, A.: Boosting cooperation by evolving trust. *Applied Artificial Intelligence* 14, 769–784 (2000)
3. Blickle, T., Thiele, L.: A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* 4(4) (1996)
4. Cheetham, W.: Case-based reasoning with confidence. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS, vol. 1898, pp. 15–25. Springer, Heidelberg (2000)
5. Cunningham, P., Zenobi, G.: Case representation issues for case-based reasoning from ensemble research. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS, vol. 2080, pp. 146–157. Springer, Heidelberg (2001)
6. Delany, S.J., Cunningham, P., Coyle, L.: An assessment of case-based reasoning for spam filtering. *Artificial Intelligence Review* 24(3), 359–378 (2005)
7. Fawcett, T.: An introduction to ROC analysis. *Pattern Recognition Letters* 27, 861–874 (2006)
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1), 119–139 (1997)
9. Hernández-Lobato, D., Hernández-Lobato, J.M., Ruiz-Torrubiano, R., Valle, Á.: Pruning adaptive boosting ensembles by means of a genetic algorithm. In: Corchado, E., Yin, H., Botti, V., Fyfe, C. (eds.) IDEAL 2006. LNCS, vol. 4224, pp. 322–329. Springer, Heidelberg (2006)
10. Jarmulak, J., Craw, S., Crowe, R.: Genetic algorithms to optimise CBR retrieval. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS, vol. 1898, pp. 136–147. Springer, Heidelberg (2000)
11. Komathyk, K., Narayanasamy, P.: Trust-based evolutionary game model assisting aodv routing againsts selfishness. *Journal of network and computer-application* 31(4), 446–471 (2008)



12. Martin, F.J., Plaza, E., Arcos, J.L.: Knowledge and experience reuse through communication among competent (peer) agents. *International Journal of Software Engineering and Knowledge Engineering* 9(3), 319–341 (1999)
13. Mitchell, M.: *An Introduction to Genetic Algorithms*. MIT Press, Cambridge (1998)
14. Muhlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary Computation* 1(1), 25–49 (1993)
15. Nagendra-Prasad, M.V., Plaza, E.: Corporate memories as distributed case libraries. In: 10th Banff Knowledge Acquisition for Knowledge-based Systems Workshop, pp. 1–19 (1996)
16. Ontañón, S., Plaza, E.: A bartering approach to improve multiagent learning. In: *Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 386–393 (2002)
17. Ontañón, S., Plaza, E.: Cooperative multiagent learning. In: Alonso, E., Kazakov, D., Kudenko, D. (eds.) *AAMAS 2000 and AAMAS 2002*. LNCS (LNAI), vol. 2636, pp. 1–17. Springer, Heidelberg (2003)
18. Pohlheim, H.: *Genetic and evolutionary algorithm toolbox for use with matlab* (1994), <http://www.geatbx.com/docu/index.html>
19. Pous, C., Gay, P., Pla, A., Brunet, J., Sanz, J., López, B.: Modeling reuse on case-based reasoning with application to breast cancer diagnosis. In: Dochev, D., Pistore, M., Traverso, P. (eds.) *AIMSA 2008*. LNCS (LNAI), vol. 5253, pp. 322–332. Springer, Heidelberg (2008)
20. Pous, C., Gay, P., Pla, A., López, B.: Collecting methods for medical CBR development and experimentation. In: Schaaf, M. (ed.) *Workshop Proceedings of the 9th European Conference on Case-Based Reasoning, CBR in the Health Sciences (ECCBR-HC)*, Trier, pp. 89–98. Tharax-Verlag (2008)
21. Russell, S., Norvig, P.: *Artificial Intelligence: A modern approach*, 2nd edn. Prentice-Hall, Englewood Cliffs (2003)
22. Santos, E.M.D., Sabourin, R., Maupin, P.: Overfitting cautious selection of classifier ensembles with genetic algorithms. *Information Fusion* 10(2), 150–162 (2009)
23. Stahl, A., Gabel, T.: Local similarity measures using evolution programs to learn. In: Ashley, K.D., Bridge, D.G. (eds.) *ICCBR 2003*. LNCS, vol. 2689, pp. 537–551. Springer, Heidelberg (2003)
24. Sun, Z., Finnier, G.R.: Case based reasoning in multiagent systems (ch. 7). In: *Intelligent techniques in E-commerce: A case-based reasoning perspective*. Springer, Heidelberg (2004)
25. Teodorescu, E.I., Petridis, M.: An architecture for multiple heterogeneous case-based reasoning employing agent technologies. In: *CIMAS (2008)*, <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-375/>
26. Tsymbal, A., Pechenizkiy, M., Cunningham, P.: Diversity in search strategies for ensemble feature selection. *Information Fusion* 6(1), 83–98 (2005)
27. Witten, I.H., Frank, E.: *Data Mining: Practical machine learning tools and techniques*, 2nd edn. Morgan Kaufmann, San Francisco (2005)
28. Yalabik, I., Yarman-Vural, F.T., Uçoluk, G., Sehitoglu, O.T.: A pattern classification approach for boosting with genetic algorithms. In: *22nd International Symposium on Computer and Information Sciences*, pp. 1–6 (2007)

# Using Meta-reasoning to Improve the Performance of Case-Based Planning

Manish Mehta, Santiago Ontañón, and Ashwin Ram

CCL, Cognitive Computing Lab  
Georgia Institute of Technology  
Atlanta, GA 30332/0280  
{mehtama1,santi,ashwin}@cc.gatech.edu

**Abstract.** Case-based planning (CBP) systems are based on the idea of reusing past successful plans for solving new problems. Previous research has shown the ability of meta-reasoning approaches to improve the performance of CBP systems. In this paper we present a new meta-reasoning approach for autonomously improving the performance of CBP systems that operate in real-time domains. Our approach uses *failure patterns* to detect anomalous behaviors, and it can learn from experience which of the failures detected are important enough to be fixed. Finally, our meta-reasoning approach can exploit both successful and failed executions for meta-reasoning. We illustrate its benefits with experimental results from a system implementing our approach called *Meta-Darmok* in a real-time strategy game. The evaluation of Meta-Darmok shows that the system successfully adapts itself and its performance improves through appropriate revision of the case base.

## 1 Introduction

Learning is a competence fundamental to intelligence, reflected in the ability of human beings to learn from their successes to make future progress and from their mistakes to improve themselves. Developing systems with learning abilities can therefore help us understand better the nature of intelligence. In order to create intelligent systems, it is thus important to provide systems with the ability to learn from their own experience (both successful and failed) and improve themselves. Failed experiences in problem solving play a role in learning, as they provide both humans and artificial systems with strong cues on what needs to be learned [5,9,10,15]. Likewise, successful experiences provide clues on ways to make progress and prosper in the problem domain. Meta-reasoning approaches have utilized successful and failed system experiences to create self-improving AI systems [16]. This paper investigates this ability of meta-reasoning approaches to improve the performance of case-based planning (CBP) systems that operate under real-time constraints.

Meta-reasoning systems are typically composed of a *base reasoner* that is in charge of the *performance task*, and a meta-reasoner that observes and modifies the base reasoner's plan. Meta-reasoning is the process of monitoring and

controlling the reasoning process of the base reasoner, to adjust that reasoning process as needed. Previous research on introspective CBR (See Section 2) has shown that meta-reasoning can enable a CBR system to learn by refining its own reasoning process. In this paper we present a new meta-reasoning approach for case-based planning. Our approach is based on using a collection of author-defined *failure patterns*, which are explicit descriptions of anomalous situations. Failure patterns provide a case-based solution for detecting failures in the execution of the base reasoner. After failures have been detected, fixes have to be made in the system to prevent those failures from happening again. Our work has four main contributions: First, it can be used in real-time domains, and instead of modifying the plans in the case-base modifies the behavior of the base system by creating *daemons* that operate in real time. Second, we present a generic representation for *failure patterns* using finite state machines (FSM). Third, our system automatically learns which of the failures detected in the behavior of the system are important or not by comparing successful and unsuccessful past executions, whereas previous approaches required an explicit model of “correct behavior” in order to detect failures. Finally, our approach can learn both from successful and unsuccessful experiences.

In this paper we are going to use Darmok [13,14] as the base reasoning system, which is a case-based planning system designed to play real-time strategy (RTS) games. The system resulting from applying our approach to Darmok is called *Meta-Darmok*, with extends Darmok by giving it the ability to be introspectively aware of successful and failed execution, analyze the differences between the two executions and further adapt and revise the executing plans based on deliberating over the analyzed differences.

The rest of the paper is organized as follows. In Section 2, we present the related approaches that use meta-reasoning to improve the performance of CBR systems. Section 3 introduces our case based planning system, Darmok, and WARGUS, the RTS game used in our experiments. In Section 5, we present our meta-reasoning approach and Meta-Darmok. We present evaluation of Meta-Darmok in Section 6. Finally we conclude with future steps in Section 7.

## 2 Related Work

Application of CBR systems to real-world problems has shown that it is difficult for developers to anticipate all possible eventualities. Changing circumstances necessitate that a CBR system learns from its experiences and improve its performance over a period of time. Fox and Leake [8] developed a system to improve the retrieval of a CBR system using meta-reasoning. Their work used a model for the correct reasoning behavior of the system, and compared the performance of the model with the actual system performance to detect and diagnose reasoning failures. The approach, however, requires the development of a complete model of correct reasoning behavior, which is difficult to construct for a system operating in a complex real-time strategy game scenario like ours. Unlike their approach, our work doesn't depend upon an author created correct model of



**Fig. 1.** A screenshot of the WARGUS game

the system’s performance but uses successful problem solving experiences as a benchmark model with which to compare the failed experience.

The DIAL system [11] uses introspective techniques to improve case adaptation. The system stores the traces of successful adaptation transformations and memory search paths and utilizes them to improve the system performance. Arcos [2] presents a CBR approach for improving solution quality in evolving environments. The approach however learns only from system’s successes whereas in our approach we utilize both the successful and failed experiences. There are other system that have explored meta-reasoning approaches to identify system failures and improve system’s performance. The Meta-Aqua system, for example, by Cox and Ram [7] uses a library of pre-defined patterns of erroneous interactions among reasoning steps to recognize failures in the story understanding task. The Autognostic system by Stroulia and Goel [16] uses a model of the system to localize the error in the system’s element and and uses the model to construct a possible alternative trace which could lead to the desired but unaccomplished solution. The Introspect system [4] observes its own behavior so as to detect its own inefficiencies and repair them. The system has been implemented for the game of Go, a deterministic, perfect information, zero-sum game of strategy between two players. Ulam et. al. [18] present a model based meta-reasoning system for FreeCiv game that uses a self-model to identify the appropriate reinforcement learning space for a specific task. All these systems however are limited to learning from failed experience (and ignoring successful experiences) to improve the system performance.

### 3 Case-Based Planning in WARGUS

WARGUS is an open source clone of WARCRAFT II, a successful commercial real-time strategy game. Each player’s goal in WARGUS is to survive and destroy the other players. Each player controls a number of troops, buildings,

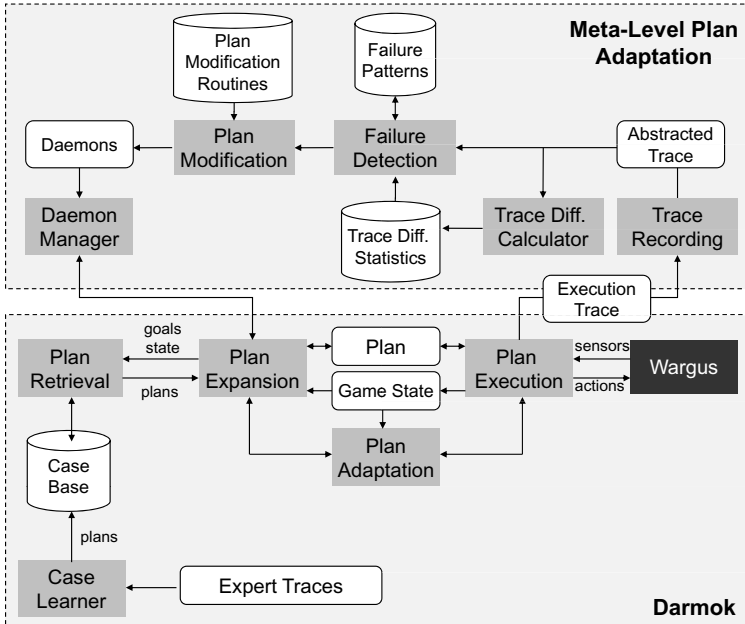


Fig. 2. Overview of the Meta-Darmok architecture

and workers (who gather resources such as gold, wood and oil in order to produce more units). Buildings are required to produce more advanced troops, and troops are required to attack the enemy. The calculations inherent in the combat system make the game non-deterministic. The game involves complex strategic reasoning, such as terrain analysis, resource handling planning, and scheduling, all of them under tight time constraints. For example, the map shown in Figure 1 is a 2-player version of the classical map “Nowhere to run, Nowhere to hide” (NWTR), one of the maps from *Battlenet* regularly played by human players, characterized by a wall of trees that separates the players. This map leads to complex strategic reasoning, such as building long range units (such as catapults or ballistas) to attack the other player before the wall of trees has been destroyed, tunneling early in the game through the wall of trees trying to catch the enemy by surprise, or other strategies, as explained in Section 6.

In this section we will briefly describe the Darmok [13, 14] system, which serves as the base reasoner for our meta-reasoning approach. In order to play WARGUS Darmok learns plans from expert demonstrations, and then uses case-based planning to play the game reusing the learnt plans. The lower part of Figure 2 shows a simplified overview of the Darmok system. Basically, Darmok is divided in two main processes:

- *Case Acquisition*: involves capturing plans for playing the game using expert demonstrations. Each time a human plays a game, an *expert trace* is

generated (containing the list of actions performed in the game). The expert then annotates the trace stating which goals he was pursuing with each action. From the annotated traces, plans are extracted and stored in the case base.

- *Plan Execution*: The execution engine consists of several modules that together maintain a current plan to win the game. The *Plan Execution* module executes the current plan, and updates its state (marking which actions succeeded or failed). The *Plan Expansion* module identifies open goals in the current plan and expands them. In order to do that it relies on the *Plan Retrieval* module, which retrieves the most appropriate plan to fulfill an open goal. Finally, the *Plan Adaptation* module adapts the retrieved plans.

Let us present some additional detail on the Darmok system, required to understand our meta-reasoning approach.

Cases in Darmok consist of two parts: *plan snippets* and *episodes*. Snippets store plans, and episodes contain information on how successful was a plan in a particular situation. Specifically, a snippet has four main parts:

- A *goal*, which is a representation of the intended goal of the plan.
- A set of *preconditions* that must be satisfied before execution.
- A set of *alive conditions* that must be satisfied during the execution of the plan for it to have chances of success.
- The plan itself.

During execution, Darmok interleaves case-based planning and execution. The plan expansion, plan execution and plan adaptation modules collaborate to maintain a current *plan* that the system is executing. A plan in Darmok is represented as a tree consisting of *goals* and *plans* (like in HTN planning [12]). Initially, the plan consists of a single goal: “win the game”. Then, the plan expansion module asks the plan retrieval module for a plan for that goal. That plan might have several subgoals, for which the plan expansion module will again ask the plan retrieval module for plans, and so on. When a goal still doesn’t have an assigned plan, we say that the goal is *open*.

Additionally, each subplan in the plan has an associated state that can be: *pending* (when it still has not started execution), *executing*, *succeeded* or *failed*. A goal that has a plan assigned and where the plan has failed is also considered to be open. Open goals can be either *ready* or *waiting*. An open goal is ready when all the plans that had to be executed before this goal have succeeded, otherwise, it is waiting.

The plan expansion module is constantly querying the current plan to see if there is any ready open goal. When this happens, the open goal is sent to the plan retrieval module. The retrieved plan is sent to the plan adaptation module, and then inserted in the current plan, marked as pending.

The plan execution module has two main functionalities: check for basic actions that can be sent to the game engine and check the status of plans that are in execution. Basic actions that are ready and with all its preconditions satisfied are sent to WARGUS to be executed. If the preconditions are not satisfied, the

action is sent back to the adaptation module to see if it can be repaired. If it cannot, then the action is marked as failed. Whenever a basic action succeeds or fails, the execution module updates the status of the plan that contained it. When a plan is marked as failed, its corresponding goal is open again. If the alive conditions of an executing plan or action are not satisfied, it is also marked as failed, and if its success conditions are satisfied, then it is marked as succeeded.

## 4 Plan Adaptation

Plan adaptation in Darmok is divided into two difference processes: parameter adaptation and structure adaptation. When a plan is retrieved from the case base, structure adaptation is used (removing or inserting actions and or goals), and when each one of the actions in a plan is about to be sent to execution, parameter adaptation is used (which will adapt the parameters of the actions: locations and unit identifiers). Let us briefly explain these two processes.

During parameter adaptation, Darmok attempts to adapt the coordinates and unit identifiers present in the actions so that they can be applied to the game at hand. For instance, if in an action the expert demonstrated that “ a farm has to be built at coordinates 26,25”, Darmok will analyze which properties the coordinates 26,25 satisfy (e.g. they represent an empty extension of grass, far from the enemy and close to a forest), and look for a location in the current map that is the most similar. Darmok will do the same with the unit identifiers (looking to use the most similar units in the current game that the expert used in his demonstration game).

For structure adaptation, Darmok analyzes a plan and determines whether all the actions are required or not in the current game. For instance, maybe a plan specifies that a “barracks” has to be built, but in the current game, we already have “barracks”. Darmok also determines whether additional actions have to be executed in order to make the plan executable in the current game. For instance maybe the plan makes reference to some “worker units” but in the current game we don’t have them, so actions in order to obtain them are required. Structure adaptation is achieved by generating a *plan dependency graph* using the preconditions and success conditions of the actions. The plan dependency graph is a directed graph where each node in the graph is a primitive action and each link represents the dependency relationship between the parent and the child.

## 5 Meta-level Plan Adaptation

The Darmok system is able to play the game of WARGUS after observing expert traces. However, once it has learnt, the system is not able to modify the plans it has learnt. Although Darmok has the capability of learning from experience (by remembering which plans succeeded and which ones failed in different situations), it does not have any capability to fix the plans in its case base. If the expert that Darmok learnt from made a mistake in one of the plans, Darmok

will repeat that mistake again and again each time Darmok retrieves that plan. The meta-reasoning approach presented in this paper provides Darmok exactly with that capability, resulting in a system called *Meta-Darmok*, shown in Figure 2. By analyzing past performances, Meta-Darmok can fix the plans in the case base, improving the performance over Darmok.

Our meta-level plan adaptation approach consists of five parts: *Trace Recording*, *Trace Difference Calculation*, *Failure Detection*, *Plan Modification*, and the *Daemon Manager*. During trace recording, a trace holding important events happening during the game is recorded. Trace difference calculation involves keeping track of differences across successful and unsuccessful games. Failure detection involves analyzing the execution trace and differences across various games to find possible issues with the executing plans by using a set of *failure patterns*. These failure patterns represent a set of pre-compiled patterns that can identify the cause of each particular failure by identifying instances of these patterns in the trace. Once a set of failures has been identified, the failed conditions can be resolved by appropriately revising the plans using a set of *plan modification routines*. These plan modification routines are created using a combination of basic modification operators (called *modops*). The modops are in the form of modifying the original elements of the plan (i.e. actions), introduction of new elements or reorganization of the elements inside the plan. Finally, some of the modifications take form of *daemons*, which monitor for failure conditions to happen when Darmok retrieves some particular behaviors. The daemon manager triggers the execution of such daemons when required. We describe each of the different parts of the meta-level plan adaptation in detail next.

## 5.1 Trace Recording

In order to analyze the behavior of Darmok, the meta-reasoning module records an *execution trace* when Darmok is playing a game. The execution trace is used to record important events happening during the execution of Darmok. These events are in the form of information regarding the plans that were executing, their start and end times, their final execution status i.e. whether the plans started, failed or succeeded.

The trace also contains the external state of the game recorded at various intervals so that different information can be extracted from it during reasoning about the failures happening at execution time. The trace provides a considerable advantage in performing adaptation of plans with respect to only analyzing the instant in which the failure occurred, since the trace can help localize portions that could possibly have been responsible for the failure.

During its run, Darmok records a low level execution trace that contains information related to basic events including the identifier of the plan that was being executed, the corresponding game state when the event occurred, the time at which the plan started, failed or succeeded, and the delay from the moment the plan became ready for execution to the time when it actually started executing. All this information is recorded in the execution trace, which the system updates as events occur at runtime. The execution trace also records any modifications



performed by the various system components to the executing goal and plans. As explained earlier, this is carried out by the plan adaptation modules. The plan adaptation module makes various changes to the list of goals and plans that are currently executing.

Once a game finishes, an *abstracted trace* is created from the execution trace that Darmok generates. The abstracted trace is the one used by the rest of components of the meta-reasoning module. The abstracted trace consists of:

- Unit Data: information regarding units such as hit points, status (whether a unit is idle, for example), location, etc.,
- Idle Data: which units were idle and the cycle intervals for which they were idle.
- Kill Data: the cycles at which particular units were attacked and killed.
- Resource Data: for each entry in the trace, the corresponding resources that were available, such as the amount of food, gold, wood and oil.
- Attack Data: the units that were involved in an attack. For example, the enemy units that were attacking and the AI units that were under attack.
- Basic Plan Failure Data: the reason for plan failures, such as whether it was due to insufficient resources or not having a particular unit available to carry out a plan.
- Trace Data: contains information including the number of goals and plans that were pursued, number that succeeded, failed, didn't finish and didn't start. The data also includes number of times a goal was pursued, number of times it restarted, the total number of resource gathering plans that were used and type of resource that was gathered by those plans as part of satisfying that goal.

Once the abstracted trace is generated, it is both sent to the *failure detection* component and to the *trace difference calculation* component. The trace difference calculator records the differences between the stored repository traces and the new execution trace as we explain next, and the failure detection component uses this information to find failures in the trace.

## 5.2 Trace Difference Calculation

Each trace records various statistics as explained previously to help calculate differences across them. The differences among other things include various statistics:

- Actions level: holds the difference in terms of the actions that are carried out. These differences are for example, differences in parameters like location where the unit is built, number of units built by an action, number and type of resources gathered.
- Non-Existent Plan/Goal: holds goals and plans that were present in a particular game and absent in other.
- Plan/Goal Type: differences in the type of plans/goals that were used during execution.

- Cycle Level: differences in start and end cycle of plans and goals.
- Goal Count: comparison of number of times a goal was pursued.
- Action Plan Stats: differences in terms of number of actions that succeeded, failed, didn't start or finish.
- Resource Gathering Plans: holds the difference in terms of number of resource gathering plans that were used and the type of resource that was gathered, the number that succeeded and failed.
- Score differences: differences in scores at the end of completion of a basic operator plan and higher level goal and plan.
- Restart differences: differences in terms of the number of times a goal was restarted.
- Plan/Goal Count: differences in terms of number of goals/plans that were used and count of particular type of plans that were used.

The *trace difference calculator* module calculates the differences for the trace pairs involving a) Type *SuccVsUnsucc*: successful vs unsuccessful traces and b) Type *UnsuccVsUnSucc*: unsuccessful vs unsuccessful traces. As Darmok plays more games of WARGUS the trace difference calculator, keeps updating the trace difference statistics. These statistics contain the probability of the differences being present across both types of trace pairs. These probability estimates are used by failure detection as explained next.

### 5.3 Failure Detection

Failure detection involves localizing the fault points. Although the abstracted trace defines the space of possible causes for the failure, it does not provide any help in localizing the cause of the failure. Traces can be extremely large, especially in the case of complex games on which the system may spend a lot of effort, thus analyzing the trace looking for failures might be a complicated problem for two reasons: first, traces might be huge, and second, it is not clear what to look for in the trace, since it is not clear what a “failure” looks like in a trace.

In order to solve both problems, our meta-reasoning module contains a collection of *failure patterns*, which can be used to identify the cause of each particular failure by identifying instances of these patterns in the trace [3,5,17]. The failure patterns essentially provide an abstraction mechanism to look for typical patterns of failure conditions over the execution trace. The failure patterns simplify the blame-assignment process into a search for instances of the particular problematic patterns. Specifically, failure detection conceptually consists of two phases, matching and filtering:

1. Failure Pattern Matching: during this phase, all the failure patterns stored in the system are matched against the abstracted trace, and all the matches are recorded.
2. Filtering Non-important Failures: during this phase, the trace difference statistics are used to discern which failures are important and which ones

can be ignored. The result of this phase is the set of failures that are passed along to the plan modification component.

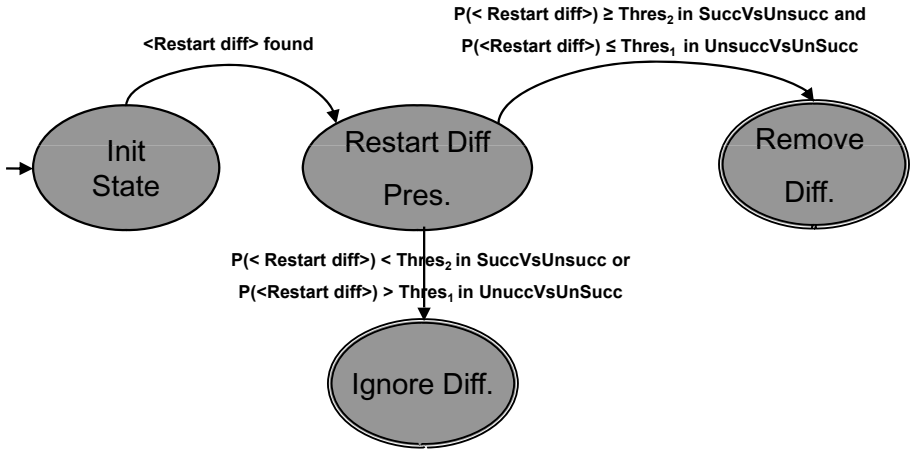
In our approach both phases are fused in a single process. Failure patterns are represented as finite state machines (FSM), which encode information needed to perform both phases mentioned earlier (matching and filtering). Figure 3 shows an example of one of the failure patterns defined in our system. Specifically, the pattern specifies that a failure is detected if a “plan restart” is found in the current trace, and if the probability of such failure satisfies some thresholds according to the trace difference statistics database.

An example of a failure detected from the abstracted trace, related to the functioning of plan adaptation module, is *Important Goals or Plans Removed failure*. The failure pattern detects whether important goals and plans have been removed from the trace. As explained earlier, the plan adaptation module potentially removes and/or adds goals and plans to existing list of goals and plans. This failure allows to detect whether, due to some failure in processing of plan adaptation module, some goals which are important for executing the current plan are removed. For example, the plan adaptation module may remove a “build tower goal” that is not considered necessary, which results in system losing in one of the maps. The fix for this failure would be to reinsert back the important goal or plan that has been removed back into the currently executing plan.

There are other failures that are detected based on the probability estimates from the trace pairs. The differences which have a high probability of presence in Type SuccVsUnsucc and low in Type UnsuccVsUnsucc are flagged for repair as part of the failure detection. This helps find the differences that are important and try to remove only those failures. These statistics help gauge probability values on whether a particular difference reflects some factor that is present or absent in successful as compared to unsuccessful games. An example of such a difference could be the *restart difference*. In Darmok, a goal restarts when it doesn't finish successfully in the previous attempt. In unsuccessful traces, it could happen that the goal restarts multiple times whereas in successful traces the goal finishes successfully in one or two attempts. This would lead to a lot of time and resources spent by the system in pursuing the restarted goal in unsuccessful traces. After analysis of the Type SuccVsUnsucc and Type UnsuccVsUnsucc traces, restart difference would have a high probability of presence in Type SuccVsUnsucc traces and low in Type UnsuccVsUnsucc. Once this difference is flagged, the failure pattern *restart failure* is detected (shown in Figure 3). A fix for this failure would be to add a constraint on the number of times the goal could be restarted. Other examples of failure patterns and their corresponding plan modification routines are given in Table 1.

## 5.4 Plan Modification

Once the cause of the failure is identified, it needs to be addressed through appropriate modification. These modifications are in the form of inserting or



**Fig. 3.** The figure shows the failure pattern *restart failure* in WARGUS. The failure is addressed when there is a high probability (greater than  $Thres_2$ ) of presence in Type *succVsUnsucc* and low probability (less than  $Thres_1$ ) in Type *UnsuccVsUnSucc*.

**Table 1.** Some example failure patterns and their associated plan modification routine in WARGUS

Failure Pattern	Plan Modification Routine
Plan Composition failure (e.g., issues with plan feature like its parameters, its type etc)	Change the plan according to the particular composition issue i.e change the location, parameter etc
Goal/Plan count Failure	Constrain the number of times a plan/goal is pursued.
Restart failure	Constrain the number of time a plan/goal is restarted
Goal/Plan Missing failure	Add a plan/goal at appropriate time during the execution
Plan Step(s) Missing failure	Add step(s) to a particular plan
Basic Operator failure	Adding a basic action that fixes the failed condition

removing a new appropriate plan at the correct position in the failed plan, or removing some steps or changing some parameter of an executing plan. The plan modification step involves applying these operators to modify the failed plans appropriately.

Once the failure patterns are detected from the execution trace, the corresponding plan modification routines and the failed conditions are inserted as daemons for the map in which these failed conditions are detected. The daemons act as a meta-level reactive plan that operates over the executing plan at runtime. The conditions for the failure pattern become the preconditions of the

plan and the plan modification routine consisting of basic modops become the steps to execute when the failure pattern is detected. The daemons operate over the executing plan, monitor their execution, detect whether a failure is about to happen and repair the plan according to the defined plan modification routines.

Notice thus, that Meta-Darmok does not directly modify the plans in the case-base of Darmok, but reactively modifies those plans when Darmok is about to use them, in case some failure is about to happen. In the current system, we have defined 20 failure patterns and plan modification routines for WARGUS.

The adaptation system can be easily extended by writing other patterns of failure that could be detected from the abstracted trace and the appropriate plan modifications to the corresponding plans that need to be carried out in order to correct the failed situation. In order to understand the process better, let us look at an example.

### 5.5 Exemplification

In order to understand the working of the meta-level plan adaptation module let us look at an illustrated example. In some runs, we observed that the plan adaptation module of Darmok inserted a large amount of resource gathering goals (in particular sending peasants to gather wood) for the following reason. In some of the expert traces, the expert sent a very small set of peasants to gather wood at the very beginning of the game. Those few peasants would gather enough wood for the rest of the game, since they would start gathering wood very early. Sometimes, Darmok would reassign one of those peasants to another task. Thus, when the need for wood arises later in the game, Darmok will be low on wood. Since wood is harvested very slowly, Darmok sends several peasants to gather wood at the same time in order to have the wood on time. This situation is fine in the short term, but in the long term, in a map like the one shown in Figure 10 (‘‘Nowhere to Run, Nowhere to Hide’’ or NWTR), a hole in the wall of trees will be opened quickly (since there are lots of peasants chopping wood). This allows the enemy to enter through the wall of trees and attack Darmok before Darmok is ready to reply. Let us see how the meta-level adaptation module can help fix this problem.

After playing several games, the trace difference calculator module accumulates statistics, and in particular one of them is that the average number of gathering resource goals for the NWTR map is much higher for unsuccessful traces than for successful traces. After the analysis of the Type SuccVsUnsucc and Type UnsuccVsUnsucc traces, the difference in statistics for the occurrence of the resource gathering goal will have a high probability of presence in Type SuccVsUnsucc traces and low in Type UnsuccVsUnsucc, thus the difference will be flagged. One of the failure patterns incorporated in our system is called *Goal/Plan Count Failure*. This failure pattern gets triggered when the difference in statistics for the occurrence of the goal or plan has a high probability of presence in Type SuccVsUnsucc traces and low in Type UnsuccVsUnsucc. When the meta-level is analyzing a new unsuccessful trace in the NWTR map, the Goal/Plan Count Failure pattern will be triggered.

The plan modification routine associated with the *Goal/Plan Count Failure* pattern consists of creating a new daemon that counts the number of times a particular goal appears during the course of a game, and if it is beyond some threshold, the daemon will delete such goals from the current plan. In particular, in our example, the daemon will be instantiated to look for resource gathering goals, and the threshold will be set to the typical value found in the successful traces in the trace difference statistics data base.

In this particular example, once the daemon was inserted, we observed that the daemon prevents addition of resource gathering goals beyond a certain point. In the short term, Darmok will struggle to have resources, since Darmok needs them to build buildings and train units. However, the daemon simply prevents the wall of trees to get destroyed. As a result resources are gathered at a slower pace and the few peasants gathering resources finally obtain enough wood, to enable Darmok to successfully complete the plan without destroying the wall of trees too early.

## 6 Meta-level Plan Adaptation Results

To evaluate Meta-Darmok, we conducted two different experiments turning the meta-reasoner on and off respectively. When the meta-reasoning is turned off, the execution traces are recorded as part of the execution. When the meta-reasoning is turned on, these execution traces are used as the starting point to calculate the trace statistics as new traces accumulate.

The experiments were conducted on 5 different variations of the NWTR map (shown in Figure 1). NWTR maps have a wall of trees separating the opponents that introduces a highly strategic component in the game (one can attempt ranged attacks over the wall of trees, or prevent the trees to be chopped by building towers, etc.). 3 different expert demonstrations were used for evaluation from NWTR maps (which lead to three different sets of plans to be used by Darmok). Moreover, Darmok can learn from more than one demonstration, so we evaluate results when Darmok learns from one, two and from all three demonstrations.

Table 2 shows the results of the experiments with and without meta-reasoning. NT indicates the number of expert traces. For each experiment 6 values are

**Table 2.** Effect of plan adaptation on game statistics

	No MetaLevel Adaptation						MetaLevel Adaptation						
<i>NT</i>	<i>W</i>	<i>D</i>	<i>L</i>	<i>ADS</i>	<i>AOS</i>	<i>WP</i>	<i>W</i>	<i>D</i>	<i>L</i>	<i>ADS</i>	<i>AOS</i>	<i>WP</i>	<i>improvement.</i>
1	4	4	7	1333	1523	<b>26.67%</b>	9	3	3	2096	582	<b>60.00%</b>	125.00%
2	5	2	8	1234	1010	<b>33.33%</b>	9	3	3	2628	356	<b>60.00%</b>	80.00%
3	5	3	7	1142	1762	<b>33.33%</b>	6	5	4	2184	627	<b>40.00%</b>	20.00%
	11	8	26	3709	4295	31.11%	24	11	10	6908	1565	53.33%	75.00%

shown: W, D and L indicate the number of wins, draws and loses respectively. ADS and AOS indicate the average Darmok score and the average opponent score (where the “score” is a number that WARGUS itself calculates and assigns to each player at the end of each game). Finally, WP shows the win percentage. The right most row presents the improvement in win percentage comparing meta-reasoning with respect to no meta-reasoning. The bottom row shows a summary of the results. The results show that meta-reasoning leads to an improvement of the percentage of wins as well as the player score to opponent score ratio. An improvement occurs in all cases irrespective of the number of traces used. Notice that in the case where Darmok learnt from 3 expert traces, the meta-reasoner is not able to bring the performance up to 60% like in the other two scenarios. This is because one of the traces used in our experiments was clearly inferior to the other two, and the performance of Darmok strongly depends on the quality of the expert traces [14]. However, meta-reasoning is still able to significantly improve the performance.

## 7 Conclusion

In this paper, we have presented a meta-reasoning approach to improve the performance of case-based planning systems, and a particular system, Meta-Darmok, that implements it. Meta-Darmok is based on the Darmok system, which plays an RTS game domain. We have shown that meta-reasoning can improve the performance of a CBP system that operates in a real-time domain. Failure patterns are useful to characterize typical failures. Moreover, by analyzing the differences between the successful and failed executions Meta-Darmok can determine which of the differences detected in failed executions with respect to successful executions are important or not. Finally, we have shown that *daemons* can be used to introduce reactive elements in the execution of the system that will adapt the behavior if failures are detected in real time. Our experimental results indicate that our approach improves the performance of the CBP system (overall improvement of 75%).

There were a few occasions when the meta-reasoning module introduced unwanted changes that degraded the system performance. However, in the few times it happened, the issue could be resolved if the system kept track of the system performance with the introduction of daemons for a particular map. If the system loses the map with the introduction of certain daemons, it could realize that the adaptation is causing unwanted changes in system performance. This might involve incorporating the actions of the meta-reasoning module into the execution trace to allow the meta-reasoner to introspect itself. We plan to explore this line in our future research. We also plan to apply our approach to other case-based planning systems to validate the generality of the approach. Finally, we also plan to investigate strategies to automatically generate failure patterns or tools for easy authoring of such failure patterns.

## References

1. Anderson, M.L., Oates, T.: A review of recent research in metareasoning and metalearning. *AI Magazine* 28, 7–16 (2007)
2. Arcos, J.L.: T-air: A case-based reasoning system for designing chemical absorption plants. In: Aha, D.W., Watson, I. (eds.) *ICCBR 2001*. LNCS, vol. 2080, pp. 576–588. Springer, Heidelberg (2001)
3. Carbonell, J.G., Knoblock, A.C., Minton, S.: *Prodigy: An Integrated Architecture for Planning and Learning*. Lawrence Erlbaum Associates, Mahwah
4. Cazenave, T.: Metarules to improve tactical go knowledge. *Inf. Sci. Inf. Comput. Sci.* 154(3-4), 173–188 (2003)
5. Cox, M.T., Ram, A.: Failure-driven learning as input bias. In: *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 231–236 (1994)
6. Cox, M.T.: Metacognition in computation: a selected research review. *Artif. Intell.* 169(2), 104–141 (2005)
7. Cox, M.T., Ram, A.: *Introspective multistrategy learning: On the construction of learning strategies*. Technical report (1996)
8. Fox, S., Leake, D.: Introspective reasoning for index refinement in case-based reasoning. *Journal of Experimental and Theoretical Artificial Intelligence* 13, 63–88 (2001)
9. Hammond, K.J.: Learning to anticipate and avoid planning problems through the explanation of failures. In: *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 556–560 (1986)
10. Kolodner, J.L.: Capitalizing on failure through case-based inference. In: *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pp. 715–726 (1987)
11. Leake, D.B., Kinley, A., Wilson, D.: Learning to improve case adaptation by introspective reasoning and CBR. In: Aamodt, A., Veloso, M.M. (eds.) *ICCBR 1995*, vol. 1010, pp. 229–240. Springer, Heidelberg (1995)
12. Nau, D., Au, T.C., Ilghami, O., Kuter, U., Wu, D., Yaman, F., Muñoz-Avila, H., Murdock, J.W.: Applications of shop and shop2. *Intelligent Systems* 20(2), 34–41 (2005)
13. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: Weber, R.O., Richter, M.M. (eds.) *ICCBR 2007*. LNCS, vol. 4626, pp. 164–178. Springer, Heidelberg (2007)
14. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: On-line case-based planning. *Computational Intelligence (to appear)*
15. Schank, R.C.: *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge University Press, Cambridge (1982)
16. Stroulia, E., Goel, A.K.: Functional representation and reasoning in reflective systems. *Journal of Applied Intelligence* 9, 101–124 (1995)
17. Sussman, J.G.: *A Computational Model of Skill Acquisition*. American Elsevier, Amsterdam (1975)
18. Ulam, P., Jones, J., Goel, A.K.: Combining model-based meta-reasoning and reinforcement learning for adapting game-playing agents. In: *AIIDE* (2008)



# Multi-level Abstractions and Multi-dimensional Retrieval of Cases with Time Series Features

Stefania Montani<sup>1</sup>, Alessio Bottrighi<sup>1</sup>, Giorgio Leonardi<sup>2</sup>, Luigi Portinale<sup>1</sup>,  
and Paolo Terenziani<sup>1</sup>

<sup>1</sup> Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria, Italy

<sup>2</sup> Dipartimento di Informatica e Sistemistica, Università di Pavia, Pavia, Italy

**Abstract.** Time series retrieval is a critical issue in all domains in which the observed phenomenon dynamics have to be dealt with. In this paper, we propose a novel, *domain independent* time series retrieval framework, based on *Temporal Abstractions* (TA). Our framework allows for *multi-level abstractions*, according to two *dimensions*, namely a taxonomy of (trend or state) symbols, and a variety of time granularities. Moreover, we allow for *flexible querying*, where queries can be expressed at any level of detail in both dimensions, also in an *interactive* fashion, and *ground cases* as well as *generalized ones* can be retrieved. We also take advantage of *multi-dimensional orthogonal index structures*, which can be refined *progressively* and *on demand*. The framework in practice is illustrated by means of a case study in hemodialysis.

## 1 Introduction

Several real world applications require to capture the evolution of the observed phenomenon over time, in order to describe its behaviour, and to exploit this information for future problem solving. In these applications, (many) process features are naturally collected in the form of time series, often automatically sampled and recorded by control instruments, as it happens e.g. in Intensive Care Unit patient monitoring [20], or in hemodialysis [17].

Case-based Reasoning (CBR) [1] is recently being recognized as a valuable knowledge management and decision support methodology in these domains, as testified by the relatively wide number of works in the field (see section 5). However, adopting CBR is typically non trivial in these situations, since the need for describing the process dynamics impacts both on case representation and on case retrieval, as analysed in [16]. In particular, similarity-based time series retrieval has to be addressed and optimized.

In the literature, most of the approaches to similarity-based time series retrieval are founded on the common premise of dimensionality reduction, which also simplifies knowledge representation (see the survey in [9]). Dimensionality is often reduced by means of a mathematical transform able to preserve the distance between two time series (or to underestimate it). Widely used transforms are the Discrete Fourier Transform (DFT) [2], and the Discrete Wavelet Transform (DWT) [7]. Another well known methodology is Piecewise Constant

Approximation (PCA) (see e.g. [12,13]), which consists in dividing a time series into  $k$  segments, and in using their average values as a  $k$ -dimensional feature vector (where obviously  $k \ll n$ , the original data dimensionality). Retrieval then works in the transformed time series space, and, with respect to the non-technical end users (e.g. the physicians), it seems to operate in a black-box fashion: the users just have to input the query, and to collect the retrieved cases, but do not (need to) see (and might not understand the meaning of) the transformed time series themselves.

In the Artificial Intelligence (AI) literature, a well known methodology is Temporal Abstractions (TA) [27,3,21,15]. TA, among the other things, have been employed for:

1. reducing time series dimensionality;
2. supporting a flexible description of phenomena at different levels of time granularity (e.g. hours, minutes, seconds);
3. providing a knowledge-based interpretation of temporal data.

Rather interestingly, TA have been scarcely explored in the CBR literature (see section 5). On the other hand, as we will extensively explain in the following, we propose to widely resort to this methodology, both for a data preprocessing step, in which time series dimensionality is reduced (see item 1 above), and as a means for supporting multiple time granularities abstractions (see item 2), at the data structure level as well as at the query level.

As regards item 1, in particular, through TA huge amounts of temporal information, like the one embedded in a time series, can be effectively mapped to a compact representation, that not only summarizes the original longitudinal data, but also abstracts meaningful behaviours in the data themselves.

Operatively, the basic principle of such TA methods is to move from a *point-based* to an *interval-based* representation of the data [3], where: the input points (*events* henceforth) are the elements of the discretized time series, and the output intervals (*episodes* henceforth) aggregate adjacent events sharing a common behaviour, persistent over time. More precisely, the method described above should be referred to as *basic* TA [3]. Basic TA can be further subdivided into *state* TA and *trend* TA. *State* TA are used to extract episodes associated to *qualitative levels* of the monitored feature, e.g. low, normal, high values; *trend* TA are exploited to detect specific *patterns*, such as increase, decrease or stationarity, in the time series. Through basic TA, a time series is therefore converted into a string of symbols, each one corresponding to an interval of raw data, and representing the (state or trend) value persistent over such an interval. Of course symbols can be mapped to intervals of different length (but a minimum time granularity is typically defined).

Despite the fact that TA are not as popular as the mathematical methodologies for reducing time series dimensionality, we believe they represent a valuable alternative with respect to more classical techniques in many domains (e.g. in medical or financial domains, in which TA methods are indeed well known), especially when: (i) a more *qualitative* abstraction of the time series values, as the one coded by abstraction symbols, is needed/sufficient; (ii) a user-friendly

mapping between raw and transformed data has to be made available. Since the output of the TA process is a sequence of symbols, usually easier to interpret for the end user with respect to the one of a mathematical transform, which would require the implementation of an additional explanatory component, we suggest that it would be useful to calculate further levels of user-interpretable abstractions over such sequence, according to two *dimensions* (see also [29]), namely: (i) a symbol taxonomy, and (ii) a time granularity taxonomy. Actually, symbols can be organized in a taxonomy, in order to provide different levels of detail in the description of episodes (of e.g. states or trends). For instance, a taxonomy of trend symbols can be introduced (see figure 1), in which the symbol  $I$  (increase) is further specialized into  $I_W$  (weak increase) and  $I_S$  (strong increase), according to the slope. On the other hand, time granularities allow one to describe episodes at different levels of temporal detail, which is the form of TA described as item 2 above. For instance, a series of three adjacent episodes of  $I$ ,  $I$  and  $S$  (stationarity), each one with a duration of 1 hour, can be merged into a single  $I$  episode, with a duration of 3 hours.

Stemming from these considerations, we are developing a *domain independent* framework for supporting time series retrieval, in which we work on cases with time series features, pre-processed by means of basic TA (henceforth *TA-based time series*), and stored in a database<sup>1</sup>. On such data, we support *multi-level abstractions*, i.e. abstractions at different detail levels according to the two dimensions outlined above. Moreover, we allow for *flexible querying*, where queries can be expressed at any level of detail in both dimensions, also in an *interactive* fashion, and *ground cases* as well as *generalized ones* (i.e. cases with features abstracted at a higher detail level, see section 3) can be retrieved. In our opinion, such flexibility and interactivity represent an additional advantage of TA-based time series retrieval with respect to more classical techniques, in which end users are unable to intervene in the retrieval process. Our framework takes advantage of *multi-dimensional orthogonal index structures*, which can be refined *progressively* and *on demand*, and which allow for early pruning and focusing during the retrieval process.

The paper is organized as follows. Section 2 introduces the data structures and functions which are needed to implement multi-level abstractions and to calculate distances for supporting flexible querying. Section 3 introduces our multi-dimensional index structures, and section 4 presents index definition and navigation algorithms, illustrating them by means of an example, taken from the hemodialysis domain. Section 5 introduces some comparisons with related work. Finally section 6 is devoted to conclusions and future work.

## 2 Data Structures and Functions for Multi-level Abstractions and Flexible Querying

As anticipated in the Introduction, we support *multi-level abstractions*, i.e. abstraction at different detail levels, according to two dimensions: (i) a taxonomy of symbols, and (ii) a taxonomy of time granularities.

<sup>1</sup> For the sake of clarity, in our description we will focus on cases with a single feature.

One of the main goals of our approach is generality: we aim at proposing a methodology that, in principle, can be applied to any domain in which time series are used and TA output is of interest. In particular, we want to allow maximal flexibility both in the description of the domain (i.e., in the taxonomy of symbols, and in the *distance* function measuring distances between symbols) and in the accuracy of temporal information (i.e., in the taxonomy of time granularities, and in the function for scaling up from a granularity to a coarser one - called *up* henceforth). On the other hand, we aim at assuring the “consistency” of the different descriptions. To do so, we have identified a set of general “consistency” constraints, that any meaningful choice must satisfy. Such constraints are motivated and illustrated below, within a description of the data structures for supporting multi-level abstractions (i.e. the taxonomies) and of their properties.

It is also worth noting that our approach allows to manage and integrate domain knowledge, when available, basically in the form of additional abstraction levels, both in dimension (i) and (ii) above (see figures 1 and 2 below for an example). However, also in absence of domain knowledge, our approach is applicable, since it can be reduced to a classical TA-based approach with one-level (i.e. flat) taxonomies in the worst case.

The **symbol taxonomy** is a conventional *isa* taxonomy that allows to describe the domain (states or trends) at increasingly more abstract levels of detail, starting from the bottom level, provided by the preprocessing TA step. An example taxonomy of symbols for trend TA is the one illustrated in figure 1. Of course, depending on the application domain, the tree can become wider or higher. The overall set of symbols in the taxonomy composes the *symbol domain*.

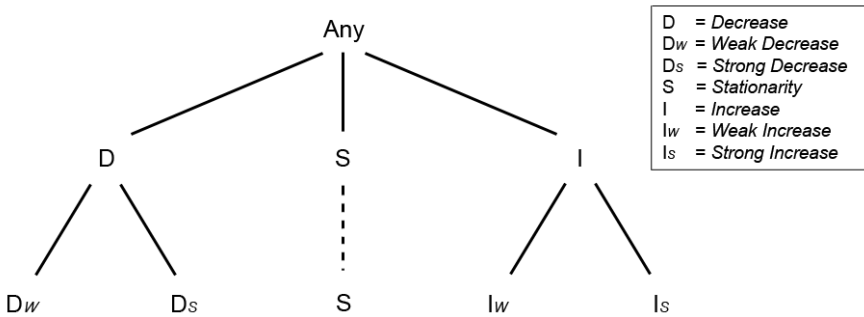


Fig. 1. An example symbol taxonomy

An important property of many symbol domains is **ordering**<sup>2</sup>. Such an ordering naturally emerges from the interpretation of the underlying row data, from which bottom symbols in the taxonomy have been abstracted. For instance,  $D_S$

<sup>2</sup> Our framework allows to treat both ordered and unordered domains. However, we will focus on ordered ones, since ordering imposes additional constraints on the domain description which are unnecessary otherwise.

(strong decrease) may abstract curve portions with slopes from  $-90$  to  $-45$  degrees, thus preceding  $D_W$  (weak decrease) (referring e.g. to slopes from  $-44$  to  $-10$  degrees), in the symbol domain ordering. Henceforth, we will use the symbols  $<^d$  and  $\leq^d$  to denote (strict) precedence in the symbol domain.

Of course, the symbol taxonomy must respect the ordering (see also [4.23](#)), if any, as stated by the following axiom:

$$\forall x, y, x', y' \in D_s \quad isa(x, x') \wedge isa(y, y') \wedge x' \neq y' \wedge x <^d y \rightarrow x' <^d y' \quad (1)$$

where  $D_s$  is the symbol domain,  $x$  is a child of  $x'$ , and  $y$  is a child of  $y'$  in the *isa* taxonomy. For instance, if  $D_W$  precedes  $I_W$  in the trend symbol ordered domain, then also  $D$  must precede  $I$ .

A **distance** function may be used in order to measure the distance between symbols in the taxonomy. As regards the distance function choice, any one can be selected. We just enforce the straightforward general constraint that the distance of each symbol from itself is zero:

$$\forall x \in D_s \quad d(x, x) = 0 \quad (2)$$

where  $D_s$  is the symbol domain, and  $d(x, x)$  denotes the distance between two identical symbols  $x$ .

While we do not impose any further constraint on the distance function for unordered symbol domains, we enforce the fact that distance must be “consistent” with ordering (if any). Specifically, distance monotonically increases with ordering, as requested by the following axiom:

$$\forall x, y, z \in D_s \quad x <^d y <^d z \rightarrow d(x, y) < d(x, z) \quad (3)$$

where  $D_s$  is the symbol domain, and  $d(x, y)$  denotes the distance between symbol  $x$  and symbol  $y$ .

For instance, referring to the trend symbol domain in figure [11](#), where the ordering is naturally given by the increasing slope values, axiom [3](#) states that the distance between  $D$  and  $S$  must be smaller than the distance between  $D$  and  $I$ .

The **granularity taxonomy**, on the other hand, allows one to describe the episodes at increasingly more abstract levels of temporal aggregation, starting from the bottom level provided by the preprocessing TA step (see figure [2](#) for an example). Obviously, the number of levels and the dimension of granules can be differently set depending on the application domain. Observe that the time dimension requires that aggregation is “homogeneous” at every given level, in the sense that each granule at a given level must be an aggregation of exactly the same number of consecutive granules at the lower level (while this number may vary from level to level; for instance, two 30 minutes long granules compose a 1 hour long granule, while three 10 minutes long granules compose a 30 minutes long granule). Such an “homogeneity” restriction is motivated by the fact that, in such a way, the duration of each episode is (implicitly) represented in the sequence of symbols. For example, at the time granularity level of 10 minutes,

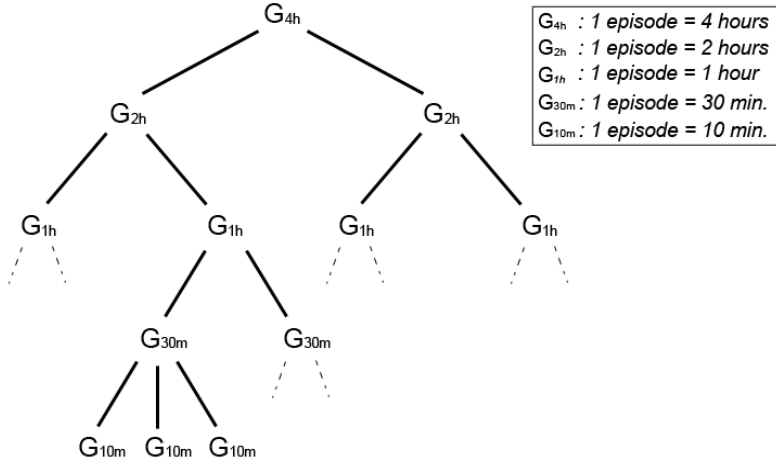


Fig. 2. An example time granularities taxonomy

the string *IIISDD* may represent a 30 minutes episode of *I* followed by 10 minutes of *S* and 20 minutes of *D*.

In order to abstract along the temporal dimension, a function for **scaling up** from one level to the coarser one in the taxonomy must be provided (called *up* henceforth). Abstracting from one granularity to a coarser one is a highly domain-dependent procedure. In order to retain the maximal generality, our framework allows one to freely define the rule. Once again, however, we impose some very general constraints, to grant for the meaningfulness of the function and for its “consistency” with respect to the other knowledge sources. The following axiom grants the fact that *up* preserves “persistence”: the result of coarsening two granules with the same symbol *x* is a larger granule still labeled as *x*. Here and in the following, for the sake of simplicity and brevity we apply the *up* function to two granules (but the definitions can be generalized to n-ary *up* operators):

$$\forall x \in D_s \quad up(x, x) = x \tag{4}$$

where  $D_s$  is the symbol domain, and  $up(x, x)$  denotes the symbol obtained by abstracting two adjacent intervals, both labelled with the same symbol *x*, at a coarser time granularity.

On the other hand, the two following axioms state the relationships between ordering and *up*, enforcing a sort of “monotonicity”: in some sense, they state that ordering is preserved by the *up* function. In particular:

$$\forall x, y \in D_s \quad x <^d y \rightarrow x \leq^d up(x, y) \leq^d y \tag{5}$$

where  $D_s$  is the symbol domain, and  $up(x, y)$  denotes the symbol obtained by abstracting two adjacent intervals, labelled with the symbols *x* and *y* respectively, at a coarser time granularity. Moreover:

$$\forall x, y, z \in D_s \quad x <^d y <^d z \rightarrow up(x, y) \leq^d up(x, z) \quad (6)$$

where  $D_s$  is the symbol domain.

Given such axioms, some unclear (or, more precisely, meaningless) situations are automatically ruled out. For instance, it can never happen that, if a 1 hour long episode of  $D$ , followed by a 1 hour long episode of  $I$ , abstracts to a 2 hours long episode of  $D$ , it also happens that a 1 hour long episode of  $D$ , followed by a 1 hour long episode of  $S$ , abstracts to a 2 hours long episode of  $S$ .

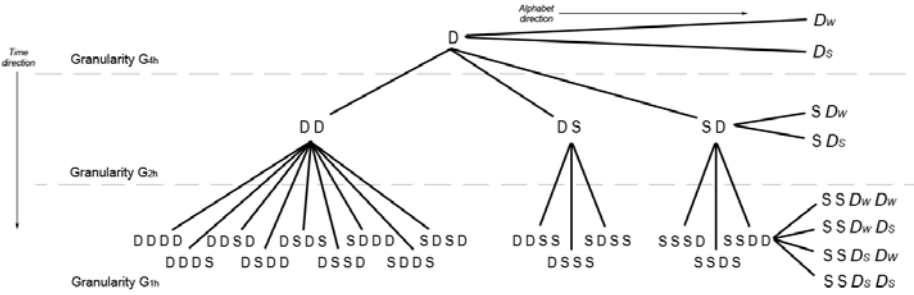
It is worth stressing that the axioms above code the relationships between the symbol ordering (if any) and the *isa* relation, the distance function, and the *up* function respectively. As a consequence, the combination of such axioms also fixes the constraints between any “combination” of such primitive notions. For instance, axioms [1](#) and [3](#) state that distance “preserves” ordering also in case *isa* relationships between symbols are involved.

### 3 Multi-dimensional Index Structures for Retrieval Optimization

Although the use of a symbol taxonomy and/or of a temporal granularity taxonomy has been already advocated in other works (e.g. in a data warehouse context, see [29](#)), to the best of our knowledge we are proposing the first approach attempting to fully exploit the advantages of taxonomical knowledge in flexible case retrieval (see section [5](#)).

Our basic idea is simple. Given the symbol taxonomy (which directly induces the abstraction function defined by the *isa* relation), the time granularity taxonomy, and the time granularity abstraction function *up*, any query can be easily abstracted at any level of symbol and/or time granularity detail (coarser than the level of the query itself). Therefore, if we provide a multi-level indexing structure addressing the different levels of abstraction, we can easily use it in order to focus our search. Starting from the most abstract level of detail, and comparing the abstracted query to the index structure nodes at progressively more accurate detail levels, our methodology can efficiently provide an early pruning of all the cases that are addressed by intermediate index layers which do not match with the query abstractions (further details on query answering will be provided in section [4](#)).

In particular, we advocate the introduction of a forest of index structures, providing a flexible indexing of cases at different levels of the symbol and/or time granularity taxonomies. The root node of each index structure is represented by a string of symbols, defined at the highest level in the symbol taxonomy (i.e. the children of “Any”, see figure [1](#)) and in the time granularity taxonomy. Potentially, a whole taxonomy of nodes can stem from each root, describing each possible refinement along the symbol and/or time granularity dimension. An example, taking as a root the  $D$  symbol, is provided in figure [3](#). Here, the root node  $D$  is refined along the time dimension from the 4 hours to the 2 hours granularity, so



**Fig. 3.** An example multi-level orthogonal index structure

that the nodes  $DD$ ,  $DS$  and  $SD$  stem from it, provided that  $up(D, S) = D$  and  $up(S, D) = D$  (see figure 3).

Moreover, we advocate that each node in each index structure belonging to the forest is itself an index, and can be defined as a *generalized case*, in the sense that it summarizes (i.e. it indexes) a set of ground cases. In the indexed cases, the feature can be abstracted as in the internal node itself (i.e. resulting in the same string), provided that we work at the same time granularity and symbol taxonomy level of the node being considered.

This means that the same ground case is typically indexed by different nodes in one index (and in the other indexes of the forest). As we will see in section 4, this supports flexible querying, since, depending on the level at which the query is issued, one of the nodes can be more suited for providing a quick answer.

Although the full generation of the forest of index structures, considering each possible level of symbol and granularity detail, is theoretically possible, for the sake of efficiency we advocate:

- the choice of a leading dimension, i.e. of a fixed order for abstractions (e.g. time granularity abstractions first, and then symbol abstractions);
- a dynamic generation/refinement of indexes, starting from a basic set of skeletal indexes which has to be defined in each specific domain/application (note that it makes sense to provide at least indexes at the coarsest symbol and time granularity levels as an initialization).

The choice of a leading dimension allows to quite naturally organize the index structure in an orthogonal way, in which, from each node of the leading dimension structure, another index stems, built according to the secondary dimension (see figure 3). In particular, the orthogonal index takes the leading index node as a root, and then progressively specializes it in the secondary dimension, keeping the leading dimension abstraction level always fixed.

It is important to stress that, although in principle the choice of a fixed order for abstractions lets our methodology loose some degree of flexibility, it does not in any way affect the expressiveness of our index structures, since, in principle, all levels of  $\langle symbol, time-granularity \rangle$  detail can be coped with (just the order in which levels are organized is affected). On the other hand, such a strategy



makes the process of abstracting queries and searching for the corresponding indexes much easier and faster, since an a priori fixed order of abstraction and search can be exploited. In particular, in figure 3 and in the rest of the paper, we have chosen time as the leading dimension.

## 4 Index Generation and Navigation

As already observed, we advocate a progressive and on-demand definition of the index structures. In particular, in the beginning it makes sense to provide a forest of trees, composed by skeletal indexes, each one rooted at a set of symbols, at the coarsest detail level, in both dimensions. Such indexes develop in the leading dimension (i.e. in time in our current approach), and are as much detailed as the domain knowledge suggests.

Further index refinement can then be automatically triggered by the types of queries which have been issued so far.

Ground queries can be answered by resorting to our abstraction mechanism and index structures. Moreover, we are able to easily treat non-ground queries as well. We just ask that all symbols in the query are at the same time granularity<sup>3</sup>.

If queries have often involved a time granularity which is not yet represented in the index(es), the corresponding level can be created. A proper frequency threshold for counting the queries has to be set to this end. We proceed analogously by creating an orthogonal index from each node which fits the frequent queries time granularity, but does not match their symbol taxonomy level.

This policy allows to augment the indexes discriminating power only when it is needed, while keeping the memory occupancy of the index structures as limited as possible.

We will now illustrate query answering in our approach, by means of an example, taken from the hemodialysis domain. In order to highlight the most innovative features of our approach, we will show an example of a non-ground query.

Hemodialysis is the most widely used treatment for End Stage Renal Disease, a severe chronic condition which, without medical intervention, leads to death. Hemodialysis relies on a device, called hemodialyzer, which clears the patient's blood from catabolites, to re-establish acid-base equilibrium and to remove water in excess. On average, hemodialysis patients are treated for four hours three times a week. Each single treatment is called a hemodialysis session, during which the hemodialyzer collects several variables, most of which are in the form of time series. Considering a case as a hemodialysis session, we want to query the case base to search for similar cases, having preprocessed the time series by means of TA.

---

<sup>3</sup> On the other hand, queries with symbols at different levels in the symbol taxonomy dimension can be easily dealt with in our approach. In particular, it is sufficient to translate every symbol at the lowest level present in the query, thus obtaining a set of queries equivalent to the original one, but easily indexable. The logic or of the single queries results has finally to be calculated.

In particular, we will focus on a single case feature, for the sake of clarity: namely, diastolic pressure. Diastolic pressure is a very powerful indicator for evaluating water reduction from the patient's blood during a session. The reduction of water from the blood during the haemodialysis session causes a constant decrease of the blood pressure. This behaviour is correct and, even if it can sometimes cause minor problems to the patient (e.g. light head spinning), it is necessary to achieve a good water and metabolites reduction. However, in certain conditions (in particular for patients suffering from cardiovascular diseases), the reduction of water is not constant, but can be characterised by stationarity periods and sudden increasing or decreasing trend episodes. In particular, problems arise when the pressure remains stationary for the most of the time (at least half of the session), which means that no water reduction takes place. Then (sharp) decreasing episodes take place, destabilising the cardiovascular system of the patient, causing problems such as faints or collapses.

An example query summarizing this negative situation is the following:  $SSD_S D_W$ , where each symbol represents a 1 hour long episode (thus globally covering the overall 4 hours duration).

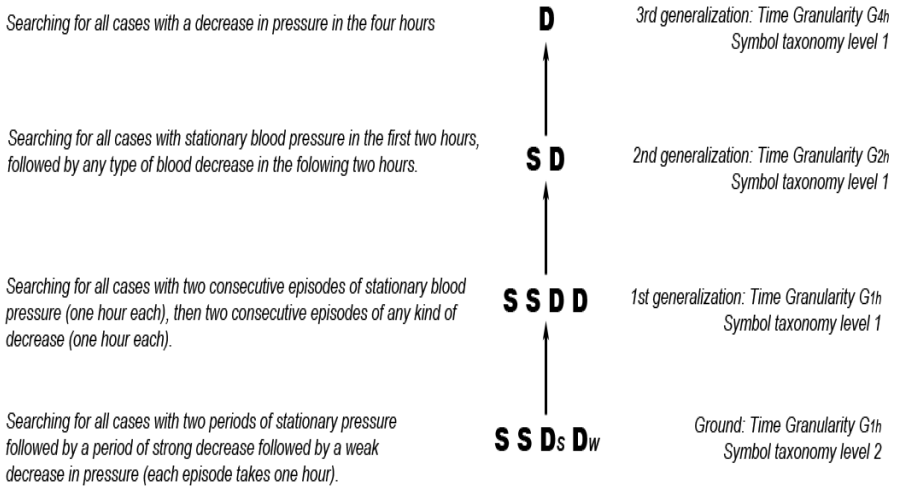
We will now show how such a query can be answered, by taking advantage of the orthogonal index structure.

Generally speaking, to answer a query, in order to enter the index structure, we first progressively generalize the query itself in the symbol taxonomy direction, while keeping time granularity fixed. Then, we generalize the query in the time dimension as well. Following the generalization steps backwards, we can enter one of the indexes in the forest from its root, and then descend along it, until we reach the node which fits the original query time granularity. If an orthogonal index stems from this node, we can descend along it, always following the query generalization steps backwards. We will stop when we reach the same detail level in the symbol taxonomy as in the original query.

If the query detail level is not represented in the index, because the index is not complete, we will stop at the most detailed possible level, which, since the abstraction order is fixed, exists and can be univocally identified. We then return all the cases indexed by the selected node.

In our example, the query generalization in the direction of the symbol taxonomy generates the sequence  $SSDD$ ; starting from the latter, the generalization in time generates the sequences:  $SD$  (2 hours long episodes), and then  $D$  (4 hours long episode). The complete generalization procedure is shown in figure 4.

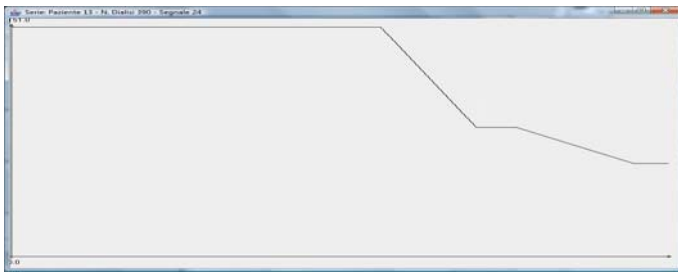
The output of the generalization process allows to identify a single index structure in the forest, namely the one whose root is  $D$  (i.e. the tree shown in figure 3) as a support for a quick query answering. Matching the steps in the generalization process to the nodes in the index structure (in the time direction), we can descend through the nodes  $SD$ , and then  $SSDD$ . Now, we can move "horizontally" in the symbol taxonomy direction, to reach the node  $SSD_S D_W$ , which matches exactly our query. As a result, we can retrieve all the cases indexed by such a node.



**Fig. 4.** Generalization steps for the diastolic pressure query

Once a set of candidate cases for a given query have been selected by navigating the index structures, distance values can be calculated by introducing any distance function which satisfies the constraints illustrated by the axioms in section 2.

In our example, among the others, the case in figure 5 is retrieved<sup>4</sup>. As the figure shows, in such a case the pressure remains constant for approximately half of the session. Then, the two decrease episodes we were searching for take place: a strong decrease followed by a weak decrease.



**Fig. 5.** Diastolic pressure in one of the retrieved cases

The query  $SSD_S D_W$  reflects a very important - but quite uncommon - situation to be investigated. Therefore, a limited number of cases are typically

<sup>4</sup> We are currently working on a real cases database, containing 1475 cases, belonging to 37 different patients.

retrieved by answering this query. We may want to generalize the required behaviour, in order to retrieve a larger number of cases. Interactive and progressive query relaxation and refinement are supported in our framework. For instance, we can allow any combination of decreasing episodes in the second half of the session. This can be obtained by relaxing the query in the direction of the symbols, using e.g. the sequence *SSDD*. A subsequent relaxation, compatible with the same set of situations, can be made in the direction of time, by using as a query the sequence *SD* (at a two hours granularity).

Query relaxation (as well as refinement) can be repeated several times, until the user is satisfied with the obtained results.

Finally, the user may want to retrieve a generalized case, i.e. to stop the search at a proper internal node in the index structure. This node subsumes a set of ground cases, but the user may just be interested in calculating the distance between the query and the node, which summarizes the retrieval set, without entering the details of all the elements composing it. For example, if the user is interested in cases with a basically stationary behaviour for the first 2 hours, and a substantially decreasing one for the following 2 hours, node *SD* in figure 3 can be retrieved in our framework.

## 5 Comparisons with Related Work

In recent years, several CBR works dealing with cases with time series features have been published, in various application domains: robot control [22], process forecast [18,24], process supervision [8], pest management [6], prediction of faulty situations [11], and medical problems [26,25,19,17]. These approaches often rely on classical mathematical dimensionality reduction techniques, such as DFT [17] and DWT [19]. Sometimes (see e.g. [25]) TA are used for data pre-processing, but basically as a noise filtering tool. Moreover, each approach has substantially been thought to support a specific application, and its generalizability is limited or not discussed at all.

A more general framework for case representation and retrieval with time dependent features has been proposed in [10]. This paper deals with the problem of time series similarity and proposes a complex retrieval strategy; we believe that our TA-based approach is more flexible, and more easily interpretable for end users. The work in [14] presents an application independent logic formalism addressing case representation when process dynamics have to be dealt with. Temporal knowledge representation for CBR is also discussed in [5]. Nevertheless, these papers do not deal with dimensionality reduction, and do not focus on retrieval solutions.

As regards TA, they have been extensively resorted to in the literature, especially in the medical field (see the survey in [28]), but typically with the aim to solve a *data interpretation task* [27] (see item 3 in the Introduction), and not as a retrieval support facility. For instance, TA have been adopted to study the co-occurrence of certain episodes in a set of clinical time series, which may justify a given diagnosis; obviously, this kind of problems are strongly based on domain knowledge, and are hardly generalizable.

Therefore, our domain independent approach for TA-based time series retrieval appears to be significantly innovative in the recent literature panorama.

It is worth noting that a database querying tool has been introduced in [29]; in this work a symbolic query (in the form of string of symbols, like the ones produced by TA) can be answered over a database of raw time series data, by producing those substrings that best match the query itself, following a set of abstraction rules, operating on a symbol taxonomy and on different time granularities. The paper thus basically introduces the same data structures we rely upon (see section 2), but exploits them only to support roll-up and drill-down operations in a data warehouse context, where the query abstraction level determines the level at which the retrieved data have to be transformed. Instead, we provide a more general and flexible retrieval support framework, in which orthogonal index structures optimize the response time, and both ground and generalized cases can be obtained. On the other hand, by now our approach operates on string matching, and not on substring matching and with the alignment problem: however, we envision such an extension as a future work.

## 6 Conclusions

In this paper, we have presented a domain independent framework for supporting time series retrieval, in which time series dimensionality is preliminarily reduced by means of TA. The use of TA provides an easily interpretable output, also for end users. Moreover, we support multi-level abstractions of TA-based time series, both along the time dimensions, and along the symbol taxonomy one, thus increasing the flexibility of the retrieval facility, especially in query definition. Queries, at various level of detail, can be made finer or coarser interactively. Query answering is also made faster by the use of orthogonal index structures, which can grow on demand. Indexes obviously allow for early pruning and focusing during the retrieval process.

In our opinion, flexibility and interactivity represent a relevant advantage of our approach to time series retrieval with respect to more classical techniques, in which end users are typically unable to intervene in the retrieval process, that often operates in a black-box fashion. In this work we have illustrated the framework in practice by means of a case study in hemodialysis. In the future, we plan to complete the framework implementation, and to extensively test the methodology by considering different domains, thus validating its significance, and studying ways of making it more and more efficient and usable.

## References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations and systems approaches. *AI Communications* 7, 39–59 (1994)
2. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: Lomet, D. (ed.) *FODO 1993*. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)

3. Bellazzi, R., Larizza, C., Riva, A.: Temporal abstractions for interpreting diabetic patients monitoring data. *Intelligent Data Analysis* 2, 97–122 (1998)
4. Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. In: Smyth, B., Cunningham, P. (eds.) *EWCBR 1998*. LNCS (LNAI), vol. 1488, p. 25. Springer, Heidelberg (1998)
5. Bichindaritz, I., Conlon, E.: Temporal knowledge representation and organization for case-based reasoning. In: *Proc. TIME 1996*, pp. 152–159. IEEE Computer Society Press, Washington (1996)
6. Branting, L.K., Hastings, J.D.: An empirical evaluation of model-based case matching and adaptation. In: *Proc. Workshop on Case-Based Reasoning, AAAI 1994* (1994)
7. Chan, K.P., Fu, A.W.C.: Efficient time series matching by wavelets. In: *Proc. ICDE 1999*, pp. 126–133. IEEE Computer Society Press, Washington (1999)
8. Fuch, B., Mille, A., Chiron, B.: Operator decision aiding by adaptation of supervision strategies. In: Aamodt, A., Veloso, M.M. (eds.) *ICCBR 1995*. LNCS (LNAI), vol. 1010, pp. 23–32. Springer, Heidelberg (1995)
9. Hetland, M.L.: A survey of recent methods for efficient retrieval of similar time sequences. In: Last, M., Kandel, A., Bunke, H. (eds.) *Data Mining in Time Series Databases*. World Scientific, London (2003)
10. Jaczynski, M.: A framework for the management of past experiences with time-extended situations. In: *Proc. ACM conference on Information and Knowledge Management (CIKM) 1997*, pp. 32–38. ACM Press, New York (1997)
11. Jaere, M.D., Aamodt, A., Skalle, P.: Representing temporal knowledge for case-based prediction. In: Craw, S., Preece, A.D. (eds.) *ECCBR 2002*. LNCS (LNAI), vol. 2416, pp. 174–188. Springer, Heidelberg (2002)
12. Keogh, E.: Fast similarity search in the presence of longitudinal scaling in time series databases. In: *Proc. Int. Conf. on Tools with Artificial Intelligence*, pp. 578–584. IEEE Computer Society Press, Washington (1997)
13. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems* 3(3), 263–286 (2000)
14. Ma, J., Knight, B.: A framework for historical case-based reasoning. In: Ashley, K.D., Bridge, D.G. (eds.) *ICCBR 2003*. LNCS (LNAI), vol. 2689, pp. 246–260. Springer, Heidelberg (2003)
15. Montani, S., Bottrighi, A., Leonardi, G., Portinale, L.: A CBR-based, closed loop architecture for temporal abstractions configuration. *Computational Intelligence* (in press)
16. Montani, S., Portinale, L.: Accounting for the temporal dimension in case-based retrieval: a framework for medical applications. *Computational Intelligence* 22, 208–223 (2006)
17. Montani, S., Portinale, L., Leonardi, G., Bellazzi, R., Bellazzi, R.: Case-based retrieval to support the treatment of end stage renal failure patients. *Artificial Intelligence in Medicine* 37, 31–42 (2006)
18. Nakhaeizadeh, G.: Learning prediction from time series: a theoretical and empirical comparison of CBR with some other approaches. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) *EWCBR 1993*. LNCS (LNAI), vol. 837, pp. 65–76. Springer, Heidelberg (1994)
19. Nilsson, M., Funk, P., Olsson, E., von Scheele, B., Xiong, N.: Clinical decision-support for diagnosing stress-related disorders by applying psychophysiological medical knowledge to an instance-based learning system. *Artificial Intelligence in Medicine* 36, 159–176 (2006)

20. Palma, J., Juarez, J.M., Campos, M., Marin, R.: A fuzzy approach to temporal model-based diagnosis for intensive care units. In: Lopez de Mantaras, R., Saitta, L. (eds.) Proc. European Conference on Artificial Intelligence (ECAI) 2004, pp. 868–872. IOS Press, Amsterdam (2004)
21. Portinale, L., Montani, S., Bottrighi, A., Leonardi, G., Juarez, J.: A case-based architecture for temporal abstraction configuration and processing. In: Proc. IEEE International Conference on Tools with Artificial Intelligent (ICTAI), pp. 667–674. IEEE Computer Society Press, Los Alamitos (2006)
22. Ram, A., Santamaria, J.C.: Continuous case-based reasoning. In: Proc. AAAI Case-Based Reasoning Workshop, pp. 86–93 (1993)
23. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proc. IJCAI, pp. 448–453 (1995)
24. Rougegrez, S.: Similarity evaluation between observed behaviours for the prediction of processes. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) EWCBR 1993. LNCS (LNAI), vol. 837, pp. 155–166. Springer, Heidelberg (1994)
25. Schmidt, R., Gierl, L.: Temporal abstractions and case-based reasoning for medical course data: Two prognostic applications. In: Perner, P. (ed.) MLDM 2001. LNCS, vol. 2123, pp. 23–34. Springer, Heidelberg (2001)
26. Schmidt, R., Heindl, B., Pollwein, B., Gierl, L.: Abstraction of data and time for multiparametric time course prognoses. In: Smith, I., Faltings, B.V. (eds.) EWCBR 1996. LNCS (LNAI), vol. 1168, pp. 377–391. Springer, Heidelberg (1996)
27. Shahar, Y.: A framework for knowledge-based temporal abstractions. *Artificial Intelligence* 90, 79–133 (1997)
28. Terenziani, P., German, E., Shahar, Y.: The temporal aspects of clinical guidelines. In: Ten Teije, A., Miksch, S., Lucas, P. (eds.) *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends* (2008)
29. Xia, B.B.: Similarity search in time series data sets. Technical report, School of Computer Science, Simon Fraser University (1997)

# On Similarity Measures Based on a Refinement Lattice

Santiago Ontañón<sup>1</sup> and Enric Plaza<sup>2</sup>

<sup>1</sup> CCL, Cognitive Computing Lab Georgia Institute of Technology,  
Atlanta, GA 30332/0280  
`santi@cc.gatech.edu`

<sup>2</sup> IIIA, Artificial Intelligence Research Institute  
CSIC, Spanish Council for Scientific Research  
Campus UAB, 08193 Bellaterra, Catalonia, Spain  
`enric@iia.csic.es`

**Abstract.** Retrieval of structured cases using similarity has been studied in CBR but there has been less activity on defining similarity on description logics (DL). In this paper we present an approach that allows us to present two similarity measures for feature logics, a subfamily of DLs, based on the concept of *refinement lattice*. The first one is based on computing the anti-unification (AU) of two cases to assess the amount of shared information. The second measure decomposes the cases into a set of independent *properties*, and then assesses how many of these properties are shared between the two cases. Moreover, we show that the defined measures are applicable to any representation language for which a refinement lattice can be defined. We empirically evaluate our measures comparing them to other measures in the literature in a variety of relational data sets showing very good results.

## 1 Introduction

Knowledge intensive case-based reasoning (CBR) has traditionally used structured representation of cases and in the recent past it has moved more close to ontology engineering and knowledge representation formalisms like description logics. Retrieval of structured cases using similarity has been studied in CBR (see section 6) but there has been less activity on defining similarity on description logics (DL) for CBR. Part of the problem is that one can define a variety of DLs: should we define a different similarity measure for each DL?

In this paper we present an approach that allows us to present two similarity measures for feature logics 8, a subfamily of DLs, based on the concept of *refinement lattice*. The concept of refinement lattice is taken from the generalization space notion of inductive learning 16, and as such is general: it is the lattice generated by a collection of refinement operators that relate two generalizations. Since any specific DL formalism can be, in principle, equipped with its own refinement operators (that induce a refinement lattice) the two similarity measures we present here can also be applied to them.



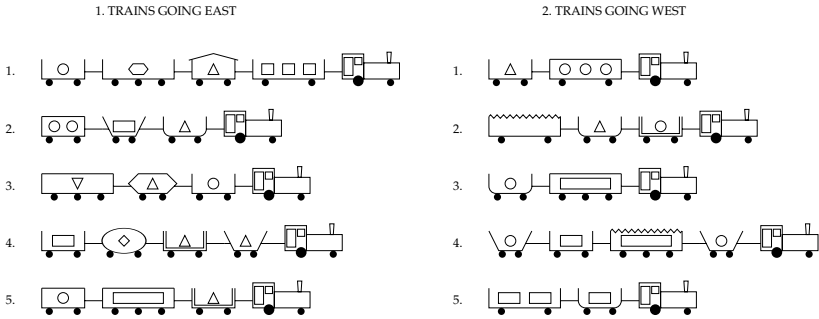


Fig. 1. Trains data set as introduced by Michalski [15]

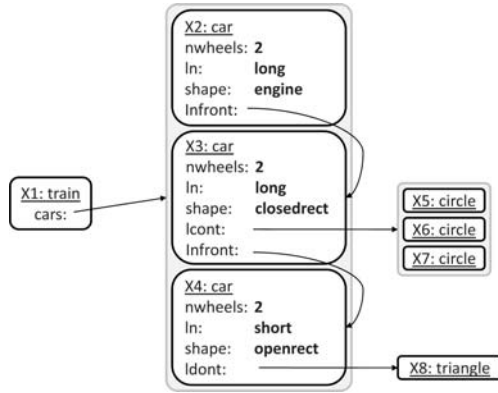
Specifically, we present two similarity measures based on a refinement lattice for feature logics. The first one computes the anti-unification (AU) of two cases, representing all the information common to these two cases; then it assesses how much information is contained in the AU with respect to the total amount of information in the two cases. We will call it *AU-based similarity* ( $S_\lambda$ ). The second measure is called *property-based similarity* ( $S_\pi$ );  $S_\pi$  decomposes the cases into a set of independent *properties*, and then assesses how many of these properties are shared between the two cases.

The remainder of this paper is organized as follows. In Section 2 we will briefly introduce some notions of relational machine learning required to define our measures. Sections 3 and 4 present the anti-unification-based measure and the property-based measure respectively. In Section 5 we describe our empirical evaluation of the measures, comparing them to other measures in the literature in a variety of relational data sets. Section 6 presents related work on relational similarity measures. Section 7 summarizes the contributions of this paper and outlines future lines of research.

## 2 A Refinement Lattice for Feature Logics

Feature logics [8] (also called feature terms, feature structures or  $\Psi$ -terms) are a generalization of first-order terms that have been introduced in theoretical computer science in order to formalize object-centered capabilities of declarative languages. In this paper we use a concrete formalization (that may differ from that of [8] or [1]), used in the language NOOS [2].

As an example, consider the apparently simple *trains* data set introduced by Michalski [15], and shown in Figure 1. The original task is to find the rule that discriminates from east-bound and west-bound trains. Notice, however, that not all the trains have the same number of cars, and that, in principle, a train can have an unbounded number of cars. Thus, it is unclear how to represent this data using a feature vector without losing information. Using a relational representation, we can just represent each car as a term, and define that a



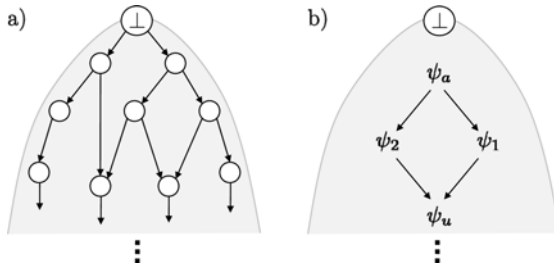
**Fig. 2.** A train represented using the feature terms representation formalism

train is a set of cars, without restricting the number of cars of the train or the complexity of the load each train is carrying. For instance, Figure 2 represents the first west-bound train using feature terms.

Feature terms can be defined by its *signature*:  $\Sigma = \langle \mathcal{S}, \mathcal{F}, \leq, \mathcal{V} \rangle$ . Where  $\mathcal{S}$  is a set of sort symbols (including  $\perp$ , which represents both the most general sort and term),  $\leq$  is a partial order among the sorts in  $\mathcal{S}$  (representing the *is-a* relation common to object oriented languages),  $\mathcal{F}$  is a set of feature symbols, and  $\mathcal{V}$  is a set of variable names. We can define a feature term  $\psi$  as:

$$\psi ::= X : s[f_1 \doteq \Psi_1, \dots, f_n \doteq \Psi_n]$$

where  $\psi$  points to the *root* variable  $X$  (that we will note as  $root(\psi)$ ),  $X \in \mathcal{V}$ ,  $s \in \mathcal{S}$ ,  $f_i \in \mathcal{F}$ , and  $\Psi_i$  might be either another feature term  $\psi_i$ , an already defined variable  $Y \in \mathcal{V}$  or a set of feature terms  $\{\psi_1, \dots, \psi_m\}$ . Finally, we also consider the basic data types (numbers and symbols) to be feature terms.



**Fig. 3.** a) Example of a *refinement lattice* defined by the subsumption relation, where each node represents a term, and the most general node is  $\perp$ . b) Example of two terms  $\psi_1$  and  $\psi_2$  in the refinement lattice with their unification  $\psi_u$  and anti-unification,  $\psi_a$ .

The basic operation between feature terms is *subsumption*, formally defined for feature terms in [2]. We say that a term  $\psi_1$  subsumes another term  $\psi_2$  when  $\psi_1$  is more general than  $\psi_2$  and we denote that as  $\psi_1 \sqsupseteq \psi_2$ . The subsumption relation allows us to structure the space of possible feature terms in a semi lattice, where the root node is  $\perp$ , also called *any*. The *refinement lattice* is defined over the semi lattice defined by subsumption in the following way: a term  $\psi_1$  is a refinement of  $\psi_2$  if  $\psi_2 \sqsubset \psi_1$  and it does not exist any  $\psi_3$  such that  $\psi_2 \sqsubset \psi_3 \sqsubset \psi_1$ .

Figure 3a shows an illustration of the refinement lattice, where each node represents a term, and arrows represent subsumption. Another interpretation of subsumption is that if a term  $\psi_1$  subsumes another term  $\psi_2$ , all the information in  $\psi_1$  is also contained in  $\psi_2$ . Typically, such space is only considered relevant for inductive learners since it defines the hypothesis space [17]. However, in this paper we are going to make use of it in order to define similarity measures.

One way to build the refinement lattice is by defining *refinement operators*. A refinement operator  $\rho$  maps a feature term to a set of feature terms that are either generalizations or specializations depending if it is a *specialization* refinement operator or a *generalization* refinement operator. Refinement operators for subsets of first order logic have been defined in the literature [16, 21]. Such refinement operators can be used in the definition of inductive systems that systematically or heuristically explore the hypothesis space.

Given the subsumption relation, and any two terms  $\psi_1$  and  $\psi_2$ , we can define the *anti-unification* of two terms as their *least general generalization* [19]. The anti-unification of two terms is relevant for defining similarity measures, since it contains all the information that is common to both  $\psi_1$  and  $\psi_2$ , thus, it encapsulates in a single description all that is common to two given terms. Moreover, depending on the representation language being used, it might not be unique (it is not unique in the case of feature terms). A complementary operation to the anti-unification is that of *unification*, which is the *most general specialization* of a given set of terms. Figure 3b graphically illustrates both concepts. Notice that both unification and anti-unification are operations over the refinement lattice: anti-unification corresponds to finding the most specific common “parent” (generalization), where as unification corresponds to finding the most general common “descendant” (specialization).

A fast algorithm to compute one of the anti-unifications of a set of terms  $T$  can be informally defined using a systematic search process over the refinement lattice in the following way. The search starts by having an initial candidate to be the anti-unification  $c_0 = \perp$ . At each step  $t$  of the algorithm, we will generate specialization refinements of the current candidate  $c_t$ . If any of those refinements subsumes all the terms in  $T$ , then that term will be taken as  $c_{t+1}$ . When in one cycle  $t$  none of the refinements subsume all of the terms in  $T$ , we will know that  $c_t$  is an anti-unification of  $T$ . Notice that this algorithm only finds one anti-unification out of all the possible ones. Moreover, the specialization refinement operator used for this algorithm must be complete, i.e. it has to be able to generate all the immediate successors of any term in the refinement lattice. It is also interesting to know the number of iterations required to find the

anti-unification of two terms (since the larger the number of steps, the larger the anti-unification, and thus the more information shared among the terms).

Notice that the previous algorithm for computing the anti-unification, although defined here for feature terms, is independent of the representation language used as long as a suitable refinement operator and subsumption operation are available. However, for completeness, next section very quickly presents a refinement operator for feature terms that can be proven to be complete (although the proof is not included in this paper for the sake of space).

## 2.1 Refinement Operators for Feature Terms

The *specialization refinement operation*  $\rho(\psi)$  for feature terms will be defined by five simpler refinement operators:  $\rho(\psi) = \rho[s](\psi) \cup \rho[f](\psi) \cup \rho[v](\psi) \cup \rho[e](\psi) \cup \rho[c](\psi)$ , as follows:

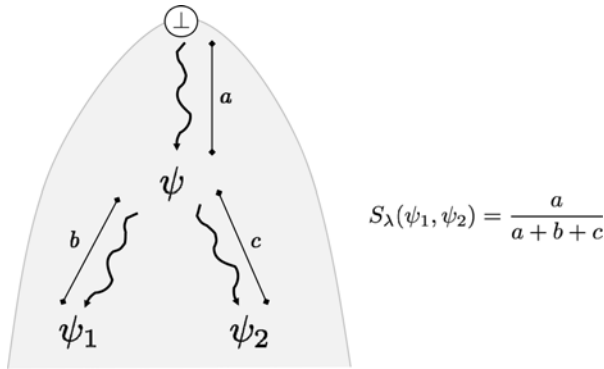
1.  $\rho[s]$  generates all the possible specializations by specializing sorts in a term.
2.  $\rho[f]$  generates specializations by taking each undefined feature in a term and adding them a variable with the most general sort that feature can take.
3.  $\rho[v]$  generates specializations by adding “variable equalities”, i.e. for any two variables  $X, Y$  in a feature term that can be unified, this operator will generate refinements where  $X = Y$ .
4.  $\rho[e]$  generates all the possible specializations by expanding any set in a feature term (including converting single values into a set of two values). The value added is the most general value allowed in that set.
5.  $\rho[c]$  generates refinements by replacing variables by constants.

Notice that in general there are an infinite number of possible refinements of a feature term (just imagine that we have a variable representing a real number, the  $\rho[c]$  operator can refine that term by substituting the variable by any concrete real number). In order to make the operator tractable, it is possible to define an alternative definition  $\rho(\psi, O)$ , where  $O$  is a set of feature terms, and only terms that subsume at least a term in  $O$  are generated. This makes the number of refinements generated always finite (e.g. the set of constants to substitute variables for can be taken from the set of constants used in  $O$ ).

Given this refinement operator, it is easy to define the dual operation  $\gamma(\psi)$ , the generalization refinement operation. The purpose of defining the specialization refinement operation is to navigate through the refinement graph. Intuitively,  $\rho(\psi)$  is an operation that maps a term to its immediate successors in the refinement lattice, and  $\gamma(\psi)$  is an operation that maps a term to its immediate ancestors (generalizations) in the lattice.

## 3 Anti-unification-Based Similarity

The anti-unification of two feature terms  $\psi_1$  and  $\psi_2$  naturally introduces a similarity measure between any two terms. The anti-unification of two terms contains



**Fig. 4.** Illustration of the anti-unification based similarity, between two feature terms  $\psi_1$ , and  $\psi_2$ , whose anti-unification is  $\psi$

all the shared information of two terms. Thus, based on that, the *anti-unification based similarity* ( $S_\lambda$ ) can be defined as: the ratio of shared information divided by the total amount of information. If two terms are very similar, the amount of common information will be very similar to the total information contained in both terms, and thus the similarity will approach 1.

We need a way to count the amount of information contained in a feature term. Using the refinement lattice, we can define the amount of information in a feature term  $\psi$  as the distance in the refinement lattice from  $\psi$  to the object  $\perp$  (the most general feature term). In other words, the number of times that a refinement operator has to be applied to  $\perp$  to generate the feature term  $\psi$ . Figure 4 illustrates this idea, where  $a$  is the number of refinement steps from  $\perp$  to the anti-unification of  $\psi_1$  and  $\psi_2$ ,  $b$  is the number of refinement steps from the anti-unification  $\psi$  to  $\psi_1$ , and  $c$  is the number of refinement steps from the anti-unification to  $\psi_2$ . Thus, we can define the similarity as:

$$S_\lambda(\psi_1, \psi_2) = \frac{a}{a + b + c}$$

Notice that by using the anti-unification algorithm outlined in Section 2, it is easy to compute  $a$  as the number of iterations required to compute the anti-unification. Moreover, in order to compute  $b$ , the same algorithm can be used, but using the anti-unification term as a starting point and taking the set  $T = \{\psi_1\}$ ;  $c$  can be computed analogously.

The resulting similarity is a simple measure that can be used to compare any two cases represented using feature terms. This measure has, however, two main issues. First of all is its computational complexity. Although computing the anti-unification of two terms requires (using the algorithm mentioned in Section 2) a linear number of calls to the subsumption operator in function of the size of the terms, the subsumption operation might have an exponential complexity depending on the representation language used. Thus, in domains where the cases in the case base are large structures, this similarity measure might not be feasible.

A second problem is that this similarity measure considers each refinement in the refinement lattice as equally important, it is like an edit distance where each operation has the same weight. Weights could be defined for each refinement operation, but it is not obvious how to generate them automatically.

## 4 Property-Based Similarity

To address the problems introduced by the anti-unification-based similarity, we developed the *property-based similarity*. In our framework, a *property* is some condition that a term might satisfy or not. For example, in the trains domain introduced before, a property might be that “a train has at least 3 cars”, and some trains might satisfy it and some might not. The main idea of the property-based similarity is to count, out of the set of properties that two cases satisfy, how many do they share.

In a feature-value representation it is easy to define the set of properties that a case satisfies: the set of features by which it is defined. However, in a complex relational representation such as feature terms, it is not obvious. In our framework, we will define a property as a pattern  $\psi_1$ , and given a term  $\psi_2$ , we say that  $\psi_2$  satisfies the property if:  $\psi_1 \sqsubseteq \psi_2$ . Therefore, the set of properties that a term satisfies is the set of all the patterns that subsume it. Notice that that set might be very large (or even infinite). Therefore, we will rely again in the notion of refinement operators to define the set of properties that a term satisfies. Each time a refinement operator is applied to a term to make it more specific, information is added to the term, and thus the term “gains a new property”. If we take the path in the refinement lattice from  $\perp$  to a particular term  $\psi$ , each one of the refinement operators in that path defines a property, for which an appropriate pattern can be constructed as explained below.

### 4.1 An Illustrative Example

Before formally explaining the process of constructing the property patterns, let us illustrate it with an example. Imagine that we have a description  $\psi$  of a train, as shown on the top of Figure 5. The train contains two cars, one of them is a long engine, and the other one is a short open rectangle car with two circles on it. Moreover, we know that the engine is in front of the open rectangle car.

If we compute the path in the refinement lattice required to reach  $\perp$  from  $\psi$  by using the  $\gamma(\psi)$  generalization refinement operator, we will see that we need 17 refinements to reach it. Each one of those 17 generalization refinements removes a piece of information from the term, and thus “removes a property”. For instance, let’s say that the first generalization takes the value *long* of the feature *ln* in the car represented by variable *X2* and generalizes it to a variable of type *length*. The property that the train has lost is that one of the cars is *long*. Thus, the first property  $\psi_1$  can be generated, as shown in Figure 5. The next generalization might generalize the value *engine* to a more general value *shape*. Leading to the second property  $\psi_2$ , that states that one of the cars of the train

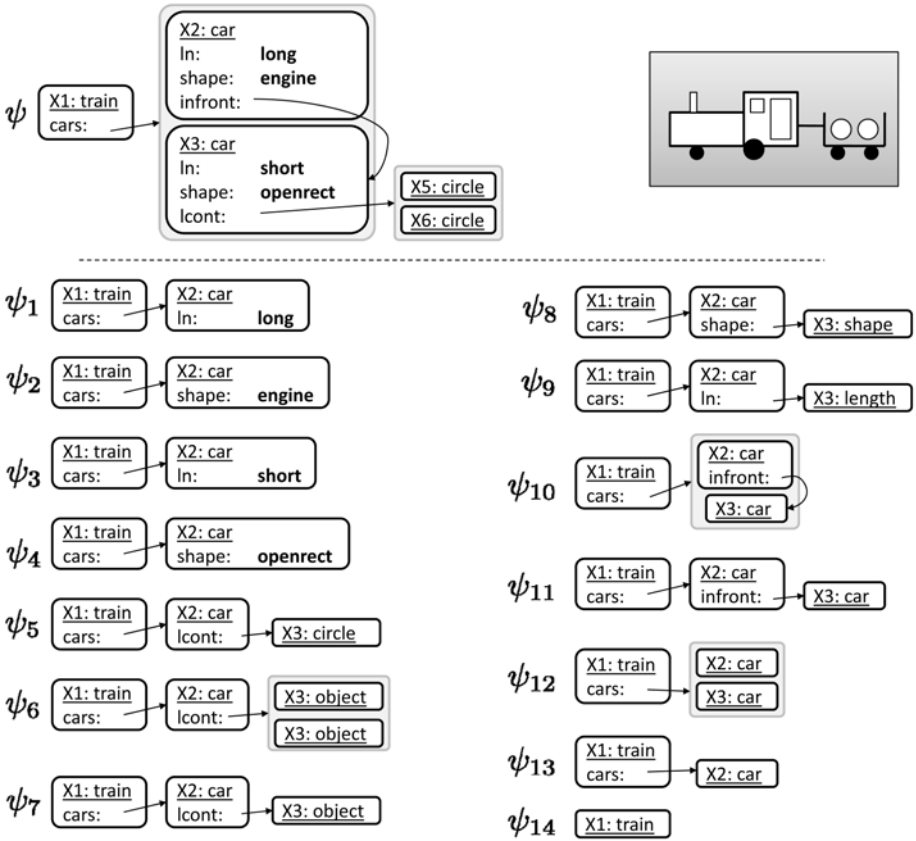


Fig. 5. A simple train represented as a feature term, and all the properties that can be extracted from it

has shape *engine*. This process can go on until we reach  $\perp$ . Figure 5 shows all the different properties that will get created in the process. Notice that there are only 14 properties in this example, but 17 refinement steps. This is because some of the properties generated result in the same pattern, and thus we have removed duplicates. Once we have the set of properties, we can use them to approximate the amount of shared information in between two cases by counting how many properties do they share.

At this point we can already see that a property associated with a generalization refinement captures exactly that piece of information that was removed from the term when generalizing. The intuition is that if we compute the unification of all the properties generated we should obtain the original term. In the case of feature terms, since unification is not unique, we can only say that one of the possible unifications of all the properties results in the original object. Therefore, the intuitive definition of a property pattern is that a pattern associated with a generalization refinement should be the smallest feature term

that if unified with the generalization allows us to reconstruct the original term. Although computing such patterns can be also done in a domain independent way using the refinement lattice, next section presents a fast way to compute them when feature terms are used as the representation language.

## 4.2 Constructing the Properties

Given a term  $\psi$ , it is possible to use the generalization refinement operator  $\gamma(\psi)$  to generalize the term step by step until  $\perp$  is reached and construct properties along the way. Notice that in order to generate the generalization refinements, subsumption is not required, and thus the process of generalizing a term until  $\perp$  is reached is not computationally expensive<sup>1</sup>. Notice that computing the shortest path from  $\psi$  to  $\perp$  might be computationally expensive, but for our similarity purposes, it is enough with finding one path (not necessarily the shortest).

Each refinement will generate a property. The generalization operator  $\gamma(\psi)$  manipulates a set of variables in a feature term in order to construct the generalizations. For instance, it might “change the sort  $s$  of a variable  $X$  to a more general sort  $s'$ ”, or “remove an element  $X$  from the feature  $f$  of another variable  $Y$ ”. In order to generate the pattern that corresponds to a property it is necessary to obtain the minimum set  $V$  of variables that constitute a path from the root of a feature term to all the variables involved in the generalization operator. For example, in the example shown in Figure 5, a generalization step might take the value *engine* of the feature *cshape* of variable  $X_2$  and generalize it by changing it to  $Y : \textit{shape}$ . For simplicity, *engine* did not have any variable name associated with it in the figure, but let us assume that its variable name is  $X_4$ . Notice that the set of involved variables are  $\{X_2, X_4\}$ . Since the root variable is  $X_1$ , the minimum set is  $V = \{X_1, X_2, X_4\}$ .

Once  $V$  has been computed, we have to compute which is the minimum set of features that each one of the variables in  $V$  require. For instance,  $X_1$  requires *cars* (since it's the only way to reach  $X_2$ ), and  $X_2$  requires *cshape*. These variables and features will be the ones appearing in the pattern associated with the property.

## 4.3 Property-Based Similarity Definition

Given a set of properties  $P$  (that can be generate by extracting them from all the cases in the case base and from the problem at hand), the first step is to compute a weight  $w_i$  for each property  $p_i \in P$ . Since each property divides the set of cases in two subsets, those which satisfy them and those which don't, a simple measure such as Quinlan's *Information Gain* [20] can be used to compute feature weights, where the weight of each feature is directly the normalized information

<sup>1</sup> One consideration has to be made when using feature terms: It is possible to construct infinite generalization chains when terms have cycles. However, by carefully selecting which generalizations to generate (basically, forbidding generalizations that increase the number of variables in a term, which are never necessary to reach  $\perp$ ), it can be proved that this problem can be completely avoided.



gain (which is the method used in our experiments to compute weights). Let us define as  $P(\psi)$  the set of properties that a particular term  $\psi$  satisfies, the similarity between two terms can be computed as:

$$S_{\pi}(\psi_1, \psi_2) = \frac{\sum_{p_i \in P(\psi_1) \cap P(\psi_2)} w_i}{\sum_{p_i \in P(\psi_1) \cup P(\psi_2)} w_i}$$

In other words, it is the sum of the weights of those properties shared by the two terms, divided by the sum of the weights of all the properties that at least one of the term satisfies. Notice, moreover, that even if the definition of the measure involves subsumption, it is actually subsumption between a property and a full term, and subsumption when one of the terms is a property is not an expensive operation. Section 5 shows a comparison in execution time between both measures, showing that the property-based measure is very efficient.

One of the main advantages of having a list of properties is that a weight can be assigned for each property. Thus, once we have two terms that we want to compare and we have extracted a set of properties, we can use information theoretical measures such as *Information Gain* [20], or the *RLDM distance* [9] to automatically assign a weight to each property.

Another interesting fact about properties, is that, under certain assumptions, if we compute the *unification* of all the properties that define a term, we obtain the original term<sup>2</sup>. In the same way, if we compute the unification of all the shared properties among a set of terms, we obtain the anti-unification of that set of terms. For that reason, if there is a single path in the refinement lattice from  $\perp$  to the anti-unification,  $S_{\lambda}$  should provide the exact same results as  $S_{\pi}$  (if uniform weights are used for the properties).

Another advantage of the property-based similarity is that its computational requirements are lower, as we will see in our experimental results section. For data sets with complex cases, computation of the anti-unification of two terms might be very costly, while properties can be extracted at a reasonable cost. The only downside of the property-based similarity, is that the anti-unification between two terms is not explicitly computed. However, it can be computed by unifying all the shared properties (at an additional computational cost). A formal evaluation of the computational complexity of both similarity measures is subject to our future work. An explicit anti-unification can be used for explanation purposes [18], as well as for adaptation purposes (since it makes explicit the similarities between a problem and the retrieved case) [7].

Finally, notice that since each property is a pattern, any other pattern generation method can be used. For instance, any relational inductive learning method that could learn descriptions that distinguish among cases in the case base could be used to generate additional patterns.

---

<sup>2</sup> This property holds only when there are no set-valued features in the involved terms.

## 5 Experimental Results

In order to evaluate our similarity measures, we used three different data sets: *sponges*, *trains*, and *kinship*. *Trains* is the data set shown in Figure 1, as presented by Michalski [15]. *Kinship* is a small but complex relational data set consisting of two families, each one with 12 members (thus 24 persons in total), proposed originally by [11], and used to evaluate several relational learning algorithms. The goal is to learn family relations. In our experiments the target relation to learn was “uncle”. The representation is purely relational, and each family is a graph (there are 4 positive examples and 20 negative examples). Finally, the *sponges* data set is a relational data set composed of 503 sponges belonging to 8 different solution classes. For the *sponges* data set, we report results both using the complete data set as well as using a subset of it (consisting of 280 sponges and 3 solution classes). We used the *trains* and *uncle* data sets as examples of data sets that are highly relational and where the value of features is not as important as the structure of the terms, and the *sponges* data set is a complex relational data set where both structure and feature values are important.

Table 1 shows the classification accuracy for several similarity measures in the data sets used for our evaluation. We report results for the two similarity measures presented in this paper, as well as two other relational similarity metrics for comparison purposes. For each similarity metric we measured classification accuracy using both a nearest neighbor as well as a 3-nearest neighbor by means of a leave-one-out method. We used SHAUD [3] and RIBL [10] (explained in detail in the next section) to compare our measures. SHAUD is a relational similarity metric defined for feature terms that has been shown to obtain very good results in complex relational data sets, and RIBL is a well known similarity measure for first order logic (FOL). RIBL requires examples to be represented in FOL and not as feature terms, but feature terms can be actually converted to FOL predicates without losing information. We used such conversion to evaluate RIBL. Moreover, RIBL and SHAUD require to know the ranges of each numeric feature before hand in order to compute similarity. We used the minimum and maximum values observed in the data set to define such ranges. Finally, RIBL requires a maximum depth parameter which was set to 10 in our experiments (large enough, since the deepest of the data sets is the *Kinship* data set where depth 5 is enough to capture each example). Finally, SHAUD only works for acyclic graphs, and thus could not be applied to the *Kinship* data set.

**Table 1.** Classification accuracy in percentage measured using a leave one out method for different similarity measures

	$S_\lambda$		$S_\pi$		SHAUD		RIBL	
	1-NN	3-NN	1-NN	3-NN	1-NN	3-NN	1-NN	3-NN
Sponges-280	95.00	94.29	<b>96.43</b>	<b>96.43</b>	95.71	95.00	91.67	91.67
Sponges-503	89.66	88.27	<b>92.25</b>	90.46	88.27	87.08	88.93	86.43
Trains-10	50.00	60.00	60.00	<b>70.00</b>	40.00	30.00	50.00	<b>70.00</b>
Kinship-24	<b>100.00</b>	91.67	<b>100.00</b>	75.00	-	-	83.33	83.33

The first thing that we can observe in Table 11 is that the property-based similarity,  $S_\pi$  achieves the highest classification accuracy in all data sets. In the Kinship data set, the only important thing is the structure. SHAUD cannot handle it since cases are cyclic graphs, and RIBL concludes that all cases have similarity 0, since they have no values in any feature (there are no numerical or symbolic values in any of the terms in Kinship, only a graph relating each member of the family to each other). Notice that RIBL achieves an accuracy of 83.33% only because it always predicts “negative”, and there are only 4 positive examples out of 24. Both  $S_\lambda$  and  $S_\pi$  are able to capture the structure of the cases, and achieve an accuracy of 100.00%. Trains is an apparently simple but complicated data set, since there are lots of features in each train, but only two are key to determine the class.  $S_\lambda$  does not compute weights for any of the differences it finds, so it cannot distinguish from differences that matter from the ones that do not matter.  $S_\pi$  and RIBL perform the best in this data set.

Finally, in the sponges data set  $S_\pi$  achieves the best results. SHAUD and  $S_\lambda$  achieve also good results but not as good, and finally RIBL gets the lowest accuracy. The problem for RIBL is that it does not exploit completely the information in the sort hierarchy, and that is important in this data set. Moreover, we would like to remark that RIBL can accept weights in both predicates and attributes, but there is no simple way to compute them directly (like with the properties in  $S_\pi$ ), and thus we used uniform weights. Thus, the results reported here for RIBL might be suboptimal, although weights won't be able to help at all in the uncle data set. Finally, we would like to note that the accuracy achieved by  $S_\lambda$  is the highest reported to date in the sponges data set.

In terms of execution time,  $S_\lambda$  takes 13.98 seconds per problem in the Sponges 503 data set,  $S_\pi$  takes 1.54 (including the time to learn the weights), SHAUD takes 4.09 seconds per problem, and RIBL is the fastest with 1.05 seconds per problem. Those times correspond to computing 502 similarities (since there are 503 examples in that data set). Time differences are similar for other data sets. We see that  $S_\lambda$  and SHAUD are the slowest since they require computing the anti-unification, and RIBL is the fastest.  $S_\pi$  is also very fast, since extracting properties does not rely on anti-unification.

We can conclude that  $S_\pi$  is the most balanced similarity overall, achieving the highest classification accuracy in most data sets while being computationally efficient. Moreover, both  $S_\lambda$  and  $S_\pi$  are conceptually very simple, and it is easy to understand what is being measured, whereas in more complex measures such as SHAUD and RIBL, it is hard to conceptually understand what exactly is being measured. Comparing  $S_\lambda$  to  $S_\pi$ ,  $S_\lambda$  has the advantage of computing an explicit symbolic similarity and of being conceptually very simple, however it is computationally expensive.  $S_\pi$  on the other hand is computationally less expensive and it is more accurate but has the disadvantages of not computing an explicit symbolic similarity term and of being conceptually more complicated (it requires the property generation step).

## 6 Related Work

Hutchinson [12] presented a distance metric based on the anti-unification of two terms. Given the anti-unification of two terms, Hutchinson measures the size of a variable substitution required to unify the anti-unification with each of the terms. The distance becomes the addition of the size of the two substitution required (for each one of the two terms we are comparing). This measure is very related to our anti-unification-based measure, but it fails to take into account some of the information, since it only counts the number of variable substitutions. For example, substituting a term *number* by *integer* or substituting it by the number 45, will count as a single substitution in Hutchinson's formalism, however, in our measure changing *number* to *integer* counts as one refinement, where as *number* to 45 requires two refinements. Thus, our measure is a more fine-grained one than the one presented by Hutchinson.

Borgida, Walsh and Hirsh [6] differentiate three generic classes of similarity measures for description logics. Our two measures fall into two of their categories.  $S_\lambda$  is what they call an *information-content based model*, and  $S_\pi$  is a *feature-based model*. They already point out that the main problem of feature-based models is identifying what constitutes a feature (a property). In this paper we have given a particular answer to that question based on refinement operators.

RIBL (Relational Instance-Based Learning) was presented by Emde and Wettschereck [10] as an approach to apply lazy learning techniques based on the nearest neighbor algorithm using first-order logic as the representation formalism. The similarity measure of RIBL uses the intuition that the similarity among two terms is the average of the similarity of the value of their features (calling this function recursively if the values are terms, thus being better suited for acyclic graphs). Moreover, they define special similarity measures if the values are numeric or symbolic. Compared to our anti-unification-based measures, RIBL has the strong point of handling naturally numerical values. However, our similarity measures are more general in the sense that we do not make any assumption about the representation language being used, but only rely on the existence of a subsumption operation and refinement operators. The fact that terms are trees, graphs or lists is irrelevant to our similarity measures. Moreover, because of the recursive way that RIBL computes similarity, values deep in the tree are bound to have less importance in the computation, where as in our property-based measure, it is left to the weight computation heuristic to decide which properties are important and which ones are not. An earlier similarity measure related to RIBL was that of Bisson [5].

An extension of the RIBL similarity measure was presented by Horváth et al [22] in order to let RIBL handle lists and terms. The extension consists of a specialized routine that uses an edit-distance to compute similarities among lists and terms added to the basic similarity measure of RIBL. The downside of the similarity metric of RIBL (including this improvement) is that specialized measures have to be defined for different type of data, where as our similarity measures can handle any kind of data uniformly.

Another approach to similarity among structured terms is that of Bergmann and Stahl [4]. They present a similarity metric specific for object oriented representations based on the concepts of *intra-class similarity* (measuring similarity among all the common features of two objects) and *inter-class similarity* (providing a maximum similarity given to object classes). The similarity is defined in a recursive way, thus limiting the approach to tree representations.

SHAUD, presented by Armengol and Plaza [3] is another similarity metric related to RIBL but designed for feature terms. SHAUD also assumes that the terms are acyclic graphs, and in the same way as RIBL and Bergmann and Stahl's it can handle numerical values in a natural way by using specialized similarity measures for different data types. Another benefit of our similarity measures with respect to RIBL and SHAUD is that it can handle comparisons among generalizations (i.e. terms that have unbound variables). Hutchinson distance can handle generalizations by using a language change representation trick mapping variables to constants, and Bergmann and Stahl define some special cases to handle this situation. Notice that this is because both similarity measures presented in this paper do not make any assumptions about the data other than assuming a subsumption relation and refinement operators.

Concerning the applicability of our measures to other formalisms, other authors have proposed refinement operators for different subsets of first-order logics or other description logics, such as Laag and Nienhuys-Cheng [14] or Shapiro [21]. Thus, making our similarity measures applicable to those representation formalisms. Moreover, feature terms can represent naturally object oriented data, making our approach applicable to those representations.

Finally, extracting properties of a term is related to the *propositionalization* operation that can map relational terms to flat feature vectors, see [13] for an overview.

## 7 Conclusions

In this paper we have presented two similarity measures for relational cases that can be used for case-based reasoning systems with complex case representations. Both similarity measures have been presented and evaluated for the feature-term representation formalism, but can be easily applied to other representation formalisms by defining an appropriate subsumption relation and refinement operators. Moreover, we have evaluated our measures with several relational data-sets showing very good results.

Compared to other similarity measures, our measures have the advantage of being independent on the representational formalism of the cases (they can work with flat feature vectors, trees, graphs, or any other if adequate refinement operators are available). The down side of the measures presented is that, due to their generality, might be computationally more expensive than other ad-hoc similarity measures, and that due to their symbolic nature, they cannot naturally handle proper comparisons among real numbers.

As part of our future work, we plan to formally evaluate the computational complexity of the measures and study ways to incorporate natural comparisons

for real-number valued data and evaluate the similarity for other representation formalisms. Other interesting lines of future work are the combination of inductive learning techniques for generating more informative patterns for a property-based similarity, and the use of the symbolic similarity and dissimilarity terms that can be computed by unifying the shared and not shared properties among two cases for different purposes such as explanation generation and adaptation.

**Acknowledgements.** Support for this work came from the project MID-CBR TIN2006-15140-C03-01.

## References

- [1] Ait-Kaci, H., Podelski, A.: Towards a meaning of life. Technical Report 11, Digital Research Laboratory (1992)
- [2] Arcos, J.L.: The Noos representation language. PhD thesis, Universitat Politècnica de Catalunya (1997)
- [3] Armengol, E., Plaza, E.: Relational case-based reasoning for carcinogenic activity prediction. *Artif. Intell. Rev.* 20(1-2), 121–141 (2003)
- [4] Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 8–13. Springer, Heidelberg (1998)
- [5] Bisson, G.: Learning in fol with a similarity measure. In: Proceedings of AAAI 1992, pp. 82–87 (1992)
- [6] Borgida, A., Walsh, T., Hirsh, H.: Towards measuring similarity in description logics. In: Horrocks, I., Sattler, U., Wolter, F. (eds.) Proceedings of the 2005 International Workshop on Description Logics (DL 2005), Edinburgh, Scotland, UK, July 26–28. CEUR Workshop Proceedings, vol. 147, CEUR-WS.org (2005)
- [7] Börner, K.: Structural similarity as a guidance in case-based design. In: Haton, J.-P., Manago, M., Keane, M.A. (eds.) EWCBR 1994. LNCS, vol. 984, pp. 197–208. Springer, Heidelberg (1994)
- [8] Carpenter, B.: Typed feature structures: an extension of first-order terms. In: Saraswat, V., Ueda, K. (eds.) Proceedings of the International Symposium on Logic Programming, San Diego, pp. 187–201 (1991)
- [9] López De Mántaras, R.: A distance-based attribute selection measure for decision tree induction. *Mach. Learn.* 6(1), 81–92 (1991)
- [10] Emde, W., Wettschereck, D.: Relational instance based learning. In: Saitta, L. (ed.) Proceedings of 13th International Conference on Machine Learning, pp. 122–130. Morgan Kaufmann Publishers, San Francisco (1996)
- [11] Hinton, G.E.: Learning distributed representations of concepts. In: Proceedings of CogSci. (1986)
- [12] Hutchinson, A.: Metrics on terms and clauses. In: van Someren, M., Widmer, G. (eds.) ECML 1997, vol. 1224, pp. 138–145. Springer, Heidelberg (1997)
- [13] Kramer, S., Lavrač, N., Flach, P.: Propositionalization approaches to relational data mining, pp. 262–286 (2000)
- [14] van der Laag, P.R.J., Nienhuys-Cheng, S.-H.: Subsumption and refinement in model inference. Technical report (1992)
- [15] Larson, J., Michalski, R.S.: Inductive inference of vl decision rules. *SIGART Bull.* (63), 38–44 (1977)

- [16] Lavrač, N., Džeroski, S.: Inductive Logic Programming. Techniques and Applications. Ellis Horwood (1994)
- [17] Mitchell, T.: Generalization as search. *Artificial Intelligence* 18(2), 203–226 (1982)
- [18] Plaza, E., Armengol, E., Ontañón, S.: The explanatory power of symbolic similarity in case-based reasoning. *Artif. Intell. Rev.* 24(2), 145–161 (2005)
- [19] Plotkin, G.D.: A note on inductive generalization. *Machine Intelligence* 5 (1970)
- [20] Quinlan, J.R.: Induction of decision trees. *Machine Learning* 1(1), 81–106 (1986)
- [21] Shapiro, E.Y.: Inductive inference of theories from facts. Technical Report 624, Department of Computer Science, Yale University (1981)
- [22] Horváth, T., Wrobel, S., Bohnebeck, U.: Relational instance-based learning with lists and terms. *Machine Learning* 43(1-2), 53–80 (2001)

# An Overview of the Deterministic Dynamic Associative Memory (DDAM) Model for Case Representation and Retrieval

Stefan Pantazi

School of Health Sciences, Community Services and Biotechnology,  
Conestoga College Institute of Technology and Advanced Learning  
299 Doon Valley Drive  
Kitchener, ON, Canada, N2G 4M4  
spantazi@conestogac.on.ca

**Abstract.** The Deterministic Dynamic Associative Memory (DDAM) is a novel associative memory model which generalizes the trie model and addresses the issues of case representation and retrieval. This paper is an overview of the DDAM model outlining its rationale, some of its design principles and its similarities with existing models and approaches. The paper will also report on a selection of experimental results.

**Keywords:** case representation, case retrieval, similarity assessment, visualization, case adaptation, CBR system design, unsupervised grammar induction, context-dependent information processing.

## 1 Introduction

Solving the closely related problems of representing and retrieving knowledge on conceptual principles in an unsupervised, human-like manner, using existing computational means, is of high interest to Artificial Intelligence (AI) and Case-Based Reasoning (CBR) research communities. The topic is also of fundamental importance to the general Computer Science community, in light of the following statement:

*“It is interesting to note that human brain is much better at secondary key retrieval than computers are; in fact, people find it rather easy to recognize faces or melodies from only fragmentary information, while computers have barely been able to do this at all. Therefore it is not unlikely that a completely new approach to machine design will someday be discovered that solves the problem of secondary key retrieval once and for all, making this entire section obsolete.” [1]*

Within this very general context, there is an acute need for intelligent approaches that can address the high complexity and sensitivity of Medical Informatics (MI) applications [2]. We have postulated this need in form of the axiom that "medical information systems must be, at the same time, usable and useful" [3]. This has supported the identification of more immediate, achievable objectives in the form of



context-dependent information processing and CBR research on associative (or content-addressable) memory models capable of unsupervised representation and retrieval of knowledge, on similarity principles [4]. The unification of these objectives was proposed in the form of the general problem of managing associative concept representation spaces characterized by four fundamental properties: high dimensionality, sparseness, dynamicity and similarity based organization [4]. Addressing each of the four fundamental properties of concept spaces led to the definition of a novel memory model, the Deterministic Dynamic Associative Memory (DDAM) model. This paper reports on this research by providing an overview of the DDAM model.

The rest of the paper is structured as follows. It begins with a background on the relevance of CBR to Medical Informatics (MI) and on the important role of Algorithmic Information Theory (AIT) in explaining this relevance at a fundamental level. The design principles of the DDAM model are discussed, followed by a characterization of the model from the perspective of the interrelated tasks of case representation and retrieval. Case representation is illustrated through the generalization of trie model and through the unsupervised grammar induction functionality. The similarity-based retrieval is illustrated by an analogy followed by experimental results. The article concludes with a description of model limitations and future work.

## 2 Background

The importance of considering similar cases in clinical decision making is very high in clinical medicine. This is why we consider Medical Informatics (MI) a case-based discipline [2] that can be defined as “context-dependent medical information processing” [3]. The high complexity, dependence on context and dynamic nature of clinical environments where often what works in one place may not work in others (spatial context-dependence) and what works today may not work tomorrow (temporal context-dependence) place serious limitations on the applicability to clinical medicine of highly distilled, general evidence derived from the study of large populations of patients. This reality differentiates MI from dominating paradigms (e.g., Biostatistics, Epidemiology, Evidence Based Medicine (EBM)) which purposely aim at removing the context around individual cases (e.g., through randomization) in order to achieve a more universal applicability of findings. As a context-dependent, case-based discipline which relies on knowledge derived from the analysis of complex descriptions of individual cases, MI is therefore a complement to population-based approaches. A more detailed discussion of the implications, including a proposal for redefining the notion of clinical evidence to account for individual case data, is available elsewhere [5].

A unifying view of scientific endeavor has provided us with some initial insights into the relevance of CBR and information retrieval to MI through the notion of “knowledge spectrum.” The relevance was further strengthened by the argument that CBR has the potential to address a fundamental issue in AI: the “frame problem” [3]. A subsequent attempt to explain at a more fundamental level why, in naturalistic environments (e.g. clinical medicine), medical decision-making is biased and often

departs from the objectiveness of normative decision models, has identified AIT as a relevant area of inquiry [3]. The tenets of AIT (e.g., algorithmic complexity, randomness, data compression, minimum description length (MDL)) have been used to define the notion of *algorithmic significance* as the mere reoccurrence (i.e., at least twice) of a sufficiently long sequence of observations (i.e., a significant pattern). Algorithmic significance may explain some of the characteristics of clinical reasoning which are based on the natural ability of recognizing patterns or regularities in data and to further the relevance of memory-based approaches, analogical reasoning and CBR to clinical medicine. In addition, because the ability of recognizing patterns or regularities in data is, in essence, a form of data compression, the opportunity to formulate a critique of the idealized MDL principle and to propose an alternative named minimum description work (MDW), was also taken. Unlike MDL, MDW is adaptable to the case of biological information processing models, where spatio-temporal complexity trade-offs seem to favor the reduction of time complexity at the expense of space complexity (i.e., memory based approaches). In terms of memory models, MDW essentially allows for a computational complexity tradeoff that aims at minimizing the read/retrieval complexity at the expense of more complex write/updates. Therefore this proposal challenges the MDL assumption that all information processing models have a limited memory that needs to be saved to the maximum extent possible through forms of extreme compression.

Finally, the MDW perspective of information processing is in agreement with the definition of “statistically rare but algorithmically significant patterns” (or algorithmic significance) where the lengths of descriptions are more important than their counts (i.e., statistics). This very idea links back to our original proposal to redefine the notion of clinical evidence to include individual case data where description lengths (i.e., algorithmic significance) are more important than their counts, as a complement to what is currently accepted as evidence by the Evidence Based medicine (EBM) movement, where counts are the sole criterion for judging significance (i.e., statistical significance).

In sum, the definition of algorithmically significant regularities or patterns, while less useful for short descriptions, becomes important for longer descriptions. This also warrants the development and evaluation of approaches that are able to discover, memorize (potentially redundantly according to the MDW rather than MDL principle) and recall efficiently the regularities whose descriptions are as long as possible and hence, as significant as possible. Functionally, such approaches are associative memories able to discover and represent algorithmically significant regularities in case data. Structurally they are compositional representations of concept spaces where significant regularities are features. Of importance to CBR, the novel memory model overviewed in this paper is a case representation and retrieval model.

### 3 The Design Principles of the DDAM Model

DDAM is a memory model capable of representing associative concept spaces characterized by four fundamental properties: high dimensionality, sparseness, dynamicity and similarity based organization. The design principles and approaches stem from the need to address each of the four fundamental properties and are summarized in Table 1.

**Table 1.** Summary of the approaches needed to address each one of the four properties of concepts spaces

Property	Approach
High dimensionality	- Hierarchical, compositional representations
Sparseness	- Dimensionality reduction (compression, grammar induction)
	- No arrays
	- Linked lists
	- Hash functions
Dynamicity	- No "ontological commitment"
	- Unsupervised, dynamic grammar induction
Similarity based organization	- No hash functions
	- Trie memory models

### 3.1 High Dimensionality

One of the most important reasons for using hierarchical and compositional approaches to overcome the high dimensionality of complex case representation is the inherent information compression capability of hierarchical models. Compositional hierarchies are often described in cognitive science, psychology and memory research literature. The structure of the medical language which is used commonly to represent clinical case data, is also hierarchical. Therefore an important structural property of the DDAM memory model must be its hierarchical nature. In fact, DDAM is a directed graph model which generalizes hierarchical representations.

In addition, the process of grammar induction, a fundamental capability of most information processors, can be regarded as a compositional approach to dimensionality reduction. The DDAM model must therefore be capable of unsupervised grammar induction.

### 3.2 Sparseness

In the context of existing computer technology, the discussion around sparseness of case representations boils down to a simple, pragmatic question: how do we represent strings of various lengths in a computer? If the strings also have a compositional structure, such as in the case of many non-random natural sequences (e.g., case descriptions), the alternative to represent them efficiently consists of linked lists which link the components that make sequences (characters, words, phrases, etc.) explicitly. Using such an approach allows representation of strings of heterogeneous, unknown lengths limited only by the available memory, while at the same time allowing for efficient update (INSERT, DELETE) functions. The fact that, in a linked list representation of a sequence, an element is used to retrieve the next element, is considered by Kanerva a good model of human memory [6]. As a result, DDAM aims at representing strings as directed graphs.

### 3.3 Dynamicity and Similarity-Based Organization

In addition to the preference for linked lists to fixed arrays, hash functions appear to be the ideal choice for addressing sparseness of data and for implementations of content addressable memories such as search and retrieval systems, content addressable memories, and other applications, which require references to information by content rather than by address. However, hash functions do not organize

information by similarity and preclude the possibility of similarity based retrieval in the manner envisioned by Kanerva. Because they do not preserve the existing structure of information, hash codes are highly artificial creations and “a poor model of human memory” [6]. Fortunately, similar results can be achieved with an associative memory model first proposed by E. Fredkin in 1960. Tries encode content information and allow for efficient similarity-based retrieval of all sequences with a given prefix or suffix. In addition to their associative property, tries are n-ary tree data structures that support efficient FIND, INSERT and DELETE operations [7]. Most importantly, in the n-ary array implementation, the time complexity of these operations does not depend on the number, but on the length of the items stored, which is of relevance to the algorithmic significance of memory content. Although efficient for dynamic applications, the n-ary trie data structure is wasteful in cases where the data is sparse and the typical number of children of each node tends to be small such as in the case of many natural sequences.

In sum, linked lists and tries are the data structure closest to the DDAM model. Similar to them DDAM provides efficient access to stored sequences, is sensitive only to the length of the alphabet on which the sequences are built, while data retrieval remains virtually independent of the number of stored sequences. The limitations of the representational power of trie models is addressed by DDAM through the use of more general structures (e.g., partial order sets, directed graphs).

### 3.4 Existing Approaches

A detailed formal description of the DDAM model is outside the scope of this paper but is available elsewhere [4]. The following is only an inventory of models, approaches and representation concepts relevant to DDAM with a focus on a few, key aspects of the model. From this perspective, DDAM can be considered:

- A generalization of the trie memory model
- A computational model for representing strings in a computer
- A directed graph
- A sequence alignment algorithm
- A unsupervised grammar induction algorithm
- A generalization of combinatorial compositions
- A deterministic memory model
- An automated indexing approach for case retrieval
- An experimental model for similarity-based retrieval

In addition, DDAM has been shown [4] to be equivalent to variable-order Markov or n-gram models and to share similarities to Self Organizing Maps (SOM), Latent Semantic Indexing (LSI) [8] and Formal Concept Analysis (FCA) [9] models as well as with connectionist models such as Kanerva’s Sparse Distributed Memory model [6]. Of interest for CBR, the DDAM model was also shown to allow for a natural mechanism for case adaptation [4] and which forms the object of future work.

The relevance of FCA to DDAM arises only from the similarity of the underlying mathematical concepts that the two share, namely that of partial order set (poset). The highly abstract nature of poset definition (a base set together with a reflexive, antisymmetric and transitive binary relation) allows such structures to be constructed in

various ways depending on the choice of the base set and binary relation. Unlike FCA where the base set is usually a collection of smaller *unordered sets* and the binary relation is the *set inclusion*, the DDAM is a finite word poset [10] based on the more restricted binary relation *is subsequence of*. This allows for the use of a base set which is a *collection of strings* (i.e., a language). The fact that in languages the order or characters is important (i.e., strings are ordered sets, tuples) implies that the DDAM partial order set has an inherent ability to represent unstructured sequential data while, at the same time, restricting the complexity of the structure. On the other hand, FCA also relies on apriori knowledge in form of existing features (e.g., attributes) required to define its highly structured formal contexts in form of tables with rows and columns whose order is immaterial. The representational power of FCA approaches based on set inclusion may be higher than that of DDAM, but FCA may also exhibit high space complexity issues that require pruning methods [11]. One fundamental question is how can one represent a case base with a few thousands of free text cases as a formal FCA context without any loss of information, i.e., having each case description fully retrievable from the context itself not from linked copies of the cases?

The choice of a restricted partial order relation in case of DDAM overcomes some of the spatial complexity while allowing for the possibility to represent fully retrievable unstructured sequential data (e.g., unstructured text) and with virtually no apriori knowledge. From this perspective, this approach to case representation is fundamentally different from existing case representation approaches based on FCA described in [12, 13] where cases are highly structured formal contexts.

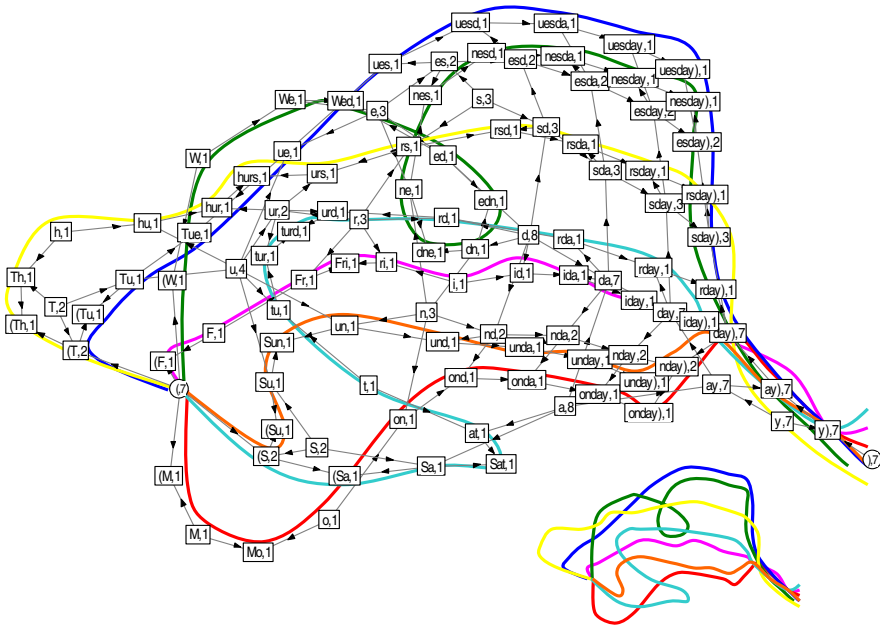
## 4 Case Representation with the DDAM Model

Semi-structured and free text case representations are common in CBR. The challenge with such representations is the automated detection of case features that one can index on in order to enable similarity based retrieval. As demonstrated further, the case representation capabilities of the DDAM are based on its ability to discover patterns and to perform unsupervised grammar induction of the case data. Therefore case data is stored as a collection of grammars induced automatically from the data.

### 4.1 The Generalized Trie Memory Model

In order to illustrate how DDAM generalizes the trie memory model, a small collection of strings (e.g., weekday names) was represented in the model. The relative small number of nodes and edges resulting from the adaptive composition algorithm on this particular data set allows us to have a complete view of the underlying structure of the memory model in Fig. 1.

In addition, force-directed automated graph layout approaches have been employed in order to obtain a more aesthetic visualization of the structure. The self-organization of the graph structure results in a similarity preserving, two-dimensional display where similar elements are closely represented. The DDAM model combined with force-directed automatic layout algorithms shares similarities with self-organizing maps [14]. This kind of similarity was also noticed by researchers on force directed automatic layout algorithms on general graphs [15].

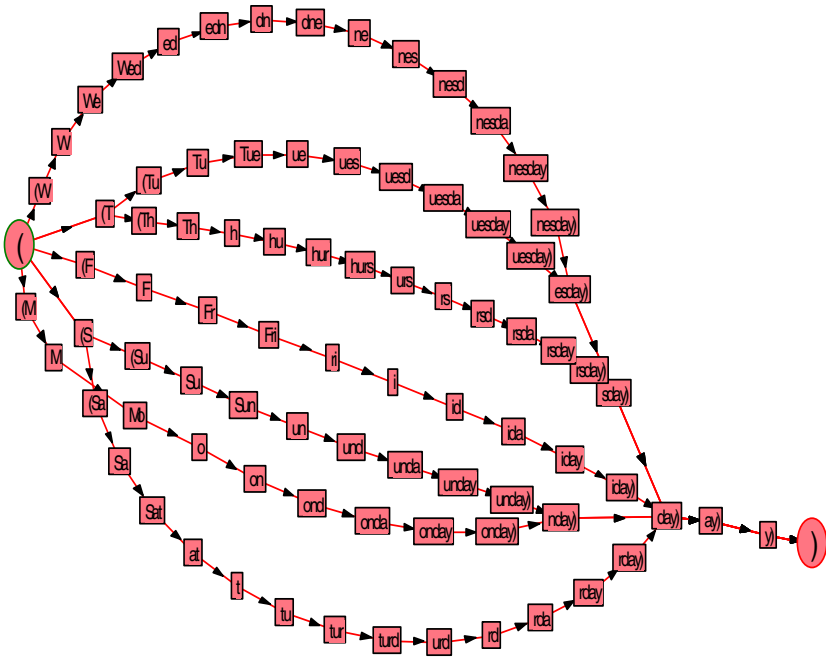


**Fig. 1.** Depiction of the compositions of the seven strings that name the seven days of week showing all the substring elements in the memory; a representation begins at a common node marked by an open round parenthesis ‘(’ and end at the ‘)’ node on the right side of the image; the string representations overlap is evident towards the end of each representation as these particular representations share common suffixes (e.g., “-day”)

Retaining only the string composition elements in the graphical display yields simpler structures such as that in Fig. 2 which is a generalization of the prefix and suffix trie structures derived from the strings in the same data set.

The structure in Fig. 2 illustrates that the representational nodes in DDAM are actually not corresponding directly to letter symbols but to more complex patterns which capture the necessary amount of context required to minimize the ambiguity of representations. For example, the 8 instances of the character *d* in the seven weekday names correspond in reality to the 8 distinct patterns *rda*, *onday*, *unday*, *iday*, *rsday*, *uesday*, *nesday* and *edn*. By capturing enough prefix and suffix context around each of the 8 *d* character instances, these more complex patterns have all become unique and have rendered the representations of weekday names non-ambiguous. The representation algorithms in DDAM aim at obtaining the same kind of representations from arbitrary unstructured sets of sequences.

The generalization of trie memory models consists of combining the prefix and suffix tries into a common data structure such as the one in Fig. 2. This generalized structure ceases to be a tree and becomes a directed graph. The DDAM model, as a generalization of the trie structure, is therefore a directed graph that is able to achieve representations of sequential data with variable levels of ambiguity that range from highly ambiguous (i.e., trivial) to non-ambiguous (i.e., optimal) representations. One of the most important features of the DDAM model is that all representations of a



**Fig. 2.** A subset of the elements in Fig. 1 showing only the elements that make the final compositions of the seven strings

sequence in DDAM are aligned and can be transformed from one into another by an adaptive string composition algorithm.

### 4.2 Unsupervised Grammar Induction

In DDAM, string compositions with certain properties correspond to formal grammars. The non-terminals in these grammars could be regarded as features that can be used to dynamically search and organize a dataset in context/content-dependent, meaningful categories. For example, in Table 2 the machine induced formal grammar of the seven strings that name the days of week (shown on the right hand side) was used to categorize the days of week according to their content (and context) by creating an inverted representation which indexes on the features (e.g., -nday, -sday). This led to the creation of a multiple hierarchy that captures similarity relationships between items and their features (e.g., Sunday and -nday) as well as between features themselves (e.g., -sday) and -esday)). Creating such multiple hierarchies is identical to the general concepts of inverted file [16] and inverted index [17] which are fundamental to information retrieval and can be successfully used in a search and retrieval on secondary keys.

**Table 2.** Example of inverted (feature indexed) multiple hierarchy derived from a machine induced formal grammar

Multiple hierarchy	Grammar rules
(T-	10 -> ( T
(Tuesday)	13 -> <u>10</u> u 11
(Thursday)	14 -> <u>10</u> h 7 12
(S-	6 -> ( S
(Saturday)	8 -> <u>6</u> a t 7 2
(Sunday)	9 -> <u>6</u> u 1
-ur-	7 -> u r
(Saturday)	8 -> 6 a t <u>7</u> 2
(Thursday)	14 -> 10 h <u>7</u> 12
-da-	4 -> d a
-day)	3 -> 4 y, 2 → 3 )
(Friday)	16 -> ( F r i <u>2</u>
(Saturday)	8 -> 6 a t 7 <u>2</u>
-nday)	1 -> n <u>2</u>
(Monday)	5 -> ( M o <u>1</u>
(Sunday)	9 -> 6 u <u>1</u>
-sday)	12 -> s <u>2</u>
(Thursday)	14 -> 10 h 7 <u>12</u>
-esday	11 -> e <u>12</u>
(Tuesday)	13 -> 10 u <u>11</u>
(Wednesday)	15 -> ( W e d n <u>11</u>

## 5 Similarity-Based Retrieval (IR) with the DDAM Model

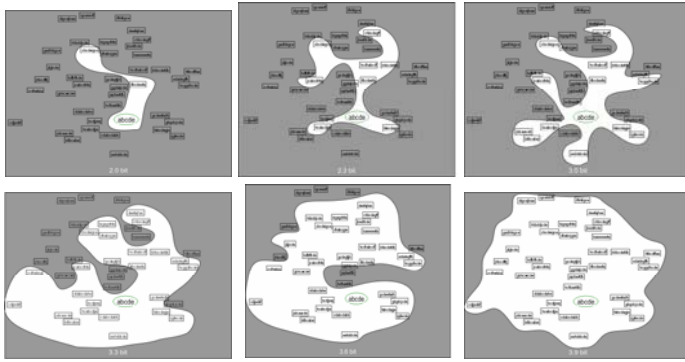
The main goal of similarity-based (associative) organization of a memory model is to enable retrieval based on similarities with a query, a process that is key to human cognition and of importance for CBR. In addition, if the retrieval is deterministic, then data must be recalled exactly. Computationally, similarity based retrieval is what Minsky and Papert referred to as “the best match problem” and, as Kanerva described it [6], it could be thought of as the retrieval of all binary descriptions stored in an associative (i.e., where similar descriptions are close), multidimensional and highly sparse binary space, within a certain bit radius of a query which is also represented as a binary description in the same space. Therefore, in a conceptual space whose organization obeys similarity principles, this kind of similarity-based retrieval would be just a read function that returns the items residing within a predefined search radius from a given query. While many case similarity measures can be defined on such a space, one of the simplest measures is just the distance in bits between the representations of two cases.

### 5.1 The “Hyperspace Telescope” Analogy

One of the best analogies for this kind of associative recall is that of a hypothetical “hyperspace telescope”, which can be centered on a query, allowing one to visualize the high dimensional similarity neighborhood of that query, within a predefined recall radius expressed in bits. The radius value corresponds to the associative recall radius that limits the number of items that can be visualized at one time with the telescope.

For example, in Fig. 3, a hypothetical hyperspace telescope with an increasingly larger associative recall radius of up to 4.0 bits was centered on the query *abcde* in the context of a collection of 10,000 strings of lengths up to 9 characters, generated





**Fig. 3.** Hypothetical hyper-space telescope with an objective radius (i.e., recall radius) ranging from 2.0 to 4.0 bits and which allows the “observation” (i.e., retrieval) of the most similar items to the query “abcde”

randomly from the alphabet {a, b, c, d, e, f, g, h, i, j}. The representation space of the randomly generated strings is highly sparse as it contains only  $10^4$  elements out of all possible  $10^9+10^8+\dots+10^1+10^0$  strings, that is, just about 0.001% of all possible strings. Through the hyperspace telescope we are able to visualize the items in the associative memory which are similar to the query, within a specified radius. The shapes of the objective of the hyperspace telescope are projections of the multi-dimensional pattern space onto a 2-dimensional display. The exact form of the objective is determined by the properties of the pattern space which, in turn, are determined by the algorithmic properties of the patterns that are represented in that space. In Table 3, the retrieved strings are shown on columns that correspond to the distance, in bits, from the query on which the hyperspace telescope was centered.

**Table 3.** The results of the DDAM associative recall on the query “abcde” on a collection of 10,000 random strings constructed from the alphabet {a, b, c, d, e, f, g, h, i, j} using an increasingly large bit radius, from 0.0 bit to 4 bit; direct similarities with the query “abcd” are shown bold and underlined, indirect associative similarities are shown in italics and underlined

0.0	2.0	2.3	3.0	3.3	3.6	3.9
<u>abcde</u>	<i>if<b>bcde</b>efa</i> <i>cb<b>de</b>igce</i>	jc <u>abcd</u> hfa hc <u>abcd</u> jje hcd <u>abcd</u> f <i>f<b>abcd</b>agje</i> ch <u>bcde</u> gff bh <u>bcde</u> fdb	eehdd <u>cde</u> ijgfe <u>cde</u> jidca <u>cde</u> hcggd <u>h</u> cde bif <u>bc</u> abei	<u>cd</u> jjediif <i>gcde</i> gfjjh <i>gcde</i> efadh <i>jcde</i> hbjig  eda <u>de</u> gffb de <u>ef</u> ajhac dh <u>abc</u> gie cd <u>abcd</u> abb dd <u>abcd</u> ahe <u>bcd</u> jjeej  bae <u>ee</u> efa bigag <u>dh</u> fa	bdfbff <u>cde</u> hhbddj <u>cde</u> jiiijj <u>cde</u> jbedif <u>cde</u> ggdaij <u>cde</u> gbgdcj <u>cde</u> gbhcai <u>cde</u>	hf <u>bcd</u> ffaa jh <u>bcd</u> ffij  <i>igce</i> ieidf <i>digce</i> jbaa <i>ifhh</i> <u>igce</u> <i>gadhh</i> <u>igce</u>  jajda <u>efdb</u> hcfba <u>efdb</u>

Upon closer inspection, a direct consequence of this kind of associative recall can be observed. Besides strings that show obvious similarities with the query (e.g., *cbcdeigce*) there are others that appear to have little in common with the original query (e.g., *igceidjf, digcejbaa*). Yet they actually do, yet not directly, but indirectly through other mediating patterns (e.g., *igce*) to which they are strongly associated. These strong associations make it possible for spurious patterns that appear to have little in common with the original query to be retrieved within the search radius. Translated to a real information retrieval situation where strings are representations of cases and where patterns in the representation are features (e.g., morphemes, words, phrases, etc.), the mechanism would allow the retrieval of documents which do not necessarily contain the query but may contain other features which are indirectly but strongly enough associated with the query (e.g., synonymous words, similar phrases, relevant contextual cues, etc.). This kind of functionality likens the DDAM model to existing approaches such as latent semantic indexing (LSI) [8], which are also known to be able to represent and recall information based on indirect associations (e.g., a query on “streptococcus” may recall documents in a biomedical collection that contain the phrase “throat infection” due to the strong association between the two).

This associative recall mechanism was inspired by Kanerva’s Sparse Distributed Memory (SDM) model [6] and is fundamental to the information retrieval capabilities of the DDAM model. At the same time, such a mechanism is extremely relevant to CBR and IR, two apparently different fields of research whose strong association was already established [18].

**5.2 A Medical Terminology Experiment**

In another experiment on representing a dataset of 1,700 compound medical terms gleaned from the various data sources, the DDAM model was queried with the term “hematoma,” which was among the terms in the dataset. The retrieved strings, shown in Table 4, are placed in columns corresponding to the concentric hyperspheres having bit radii that range from 0.0 bits to 2.0 bits. The similarity based retrieval mechanism shows among others, at 2.0 bits, an indirect association which causes terms such as *hydroxymyristoyl* which appears in the results set because of the similarity with *peristalsis* which, in turn, has obvious similarities with a close match (namely *peristomal*) to the original query *hematoma*.

**Table 4.** The results of the DDAM associative recall on the query “hematoma” on a collection of 1,700 medical compound terms within a bit radius ranging from 0.0 to 2.0 bits

0.0	1.0	1.6	2.0
hematoma	hematomas	angioedema	hemangiomas
	cephalhaematoma		hemangioma
	hematuria		angiofibriomas
	lymphohematopoietic		hematopoietic
	hepatoma		manometric
	hepatomegaly		prematurity
	peristomal		premature
			perisplenitis
			peristalsis
			hydroxymyristoyl

It is therefore clear that this approach to similarity based retrieval is bound to yield result sets that contain spurious hits caused by the combination of indirect associative recall and the potentially significant sparseness of the pattern space. However, as it was demonstrated in [4], the spurious hits can be weeded out by simply representing additional information into the model. Essentially, representing new information in the model fills some of the pattern space gaps between a query and the representations that are marginally or indirectly similar to it and which cause spurious results. To use the hyperspace telescope analogy, adding additional information in the memory effectively “knocks out” spurious elements further from the centre of the telescope objective and makes them retrievable only at higher bit radii.

To support this idea an additional experiment was done on an extended dataset which included the following terms: *stomach*, *perist*, *stomatitis*, *anastomosis*. As a result, because the new terms share similarities with the spurious result *peristomal*, on the very same query *hematoma*, the term *peristomal* which was initially retrieved within a 1.0 bit radius, was effectively “pushed” outwards from the query in the conceptual space and was retrievable only at 2.0 bit radius. Therefore, by adding relevant additional information into the system, we were able to overcome problematic associations and weed out spurious results. To generalize, this supports the idea that similarity-based retrieval can be improved incrementally and continuously by dynamically adding relevant information into a system, a property which is in perfect agreement with the functional principles of CBR and of significant importance to Medical Informatics (MI).

### 5.3 Unsupervised Equivalence Set Induction

The DDAM capabilities have also been tested in equivalence set induction experiments [4]. These experiments have culminated with the discovery of one equivalence set of extraordinary *algorithmic significance*. In terms of retrieval by similarity from case bases with complex descriptions (e.g., textual) such occurrences would be also highly unusual, unless some cases exist in multiple very similar versions.

The data sources used in the following experiments are the MedTest collection, a collection of 75 queries and 2,344 documents used for evaluation of the SAPHIRE information retrieval system [19]. Each document contains an abstract and metadata (title, authors, journal, and MeSH index terms). The collection was originally created for the evaluation of the MEDLINE system in clinical settings, and was later adapted for the evaluation of retrieval systems in biomedicine.

In an experiment, DDAM formed an equivalence set containing two distinct strings that shared extremely long identical subsequences. The sheer length of these regularities implied that their frequency to appear by chance in two distinct documents is extremely low. This occurrence is therefore an extraordinary event, with high algorithmic significance and which is highly indicative of only one, virtually unequivocal scenario: the text must have been copied from one document into the other in some way. A search on this paragraph in the original MedTest collection turned out two distinct abstracts with the identification numbers 803 and 1972, respectively. The subsequent inspection of the metadata revealed that the documents have been written by the exact same authors but published in different years in two different journals.

## 5.4 Limitations and Future Work

The main limitation of the DDAM model is related to its scalability. This is so because the memory (i.e., space) requirements are, in the worst case, a quadratic function of the length of represented sequences. However, DDAM was implemented and ran successfully on systems with only 2 gigabytes of RAM, demonstrating acceptable performance in tasks involving the MedTest collection, which is the equivalent of a case base with about 2,400 free text case descriptions.

We have also identified an immediate need for short-term applications that can overcome scalability limitations and truly demonstrate the potential of the DDAM model approach. In this regard, recent development in distributed and grid computing are of interest in pursuing. An example of immediate demonstration of the usefulness of the DDAM model is in the area of user interface design and whose efficiency could be improved through a “suggest as you type” approach that can go beyond the usual limitations of the prefix trie approach.

An additional limitation of the DDAM stems from the fact that the more distant two patterns in a sequence, the more difficult for this model to explicitly capture and represent the associations between them. Though overcoming this limitation was already envisioned through additional innovations, these will most likely increase the spatial and temporal complexity of the model.

Another important avenue of future research is the extension of the DDAM model to the representation of cases that contain multimedia and biosignal data. The rationale for this is based on the observation that the representations of 1-dimensional sequences in DDAM are 2-dimensional combinatorial objects named Dyck paths. Generalizing, it appears that in order to achieve DDAM representations of objects with a certain dimensionality, one might have to devise models that use representations whose dimensionality is necessarily higher. For example, in order to process 2-dimensional objects such as images, the representation model may have to be extended to employ 3-dimensional as “Dyck surfaces.” Though Dyck surfaces are hypothetical combinatorial objects that nobody described yet, their computational complexity would necessarily be higher than that of the images they represent but not excessively higher if data were sufficiently sparse. In sum, there appears to be little reasons that preclude the information processing model developed in this dissertation to be extended to processing representations whose dimensionality is higher than that of 1-dimensional sequences.

Finally, an important objective of this research is to continue the exploration of advanced case adaptation and retrieval that works on conceptual principles. Such similarity based retrieval approach has the potential to allow for queries such as “retrieve the most similar/relevant cases and explain their similarities to the following case description, in the following context/scenario” a capability of extreme importance in the context of CBR and the emerging Electronic Health Records.

## Acknowledgements

The author wishes to acknowledge the useful comments of the reviewers, in particular those of reviewer #4.

## References

1. Knuth, D.E.: Retrieval on Secondary Keys. In: *The art of computer programming: Sorting and Searching*, pp. 392–559. Addison-Wesley, Reading (1997)
2. Pantazi, S.V., Arocha, J.F., Moehr, J.R.: Case-based Medical Informatics. *BMC Journal of Medical Informatics and Decision Making* 4(1) (2004)
3. Pantazi, S.V., Kushniruk, A., Moehr, J.R.: The usability axiom of medical information systems. *International Journal of Medical Informatics* 75(12), 829–839 (2006)
4. Pantazi, S.V.: A Deterministic Dynamic Associative Memory (DDAM) Model for Concept Space Representation, in *School of Health Information Science*, p. 366. University of Victoria, Victoria (2006), <http://hi.conestogac.on.ca/files/dissertation-final.pdf> (accessed March 6, 2009)
5. Pantazi, S.V., Bichindaritz, I., Moehr, J.R.: The case for context-dependent dynamic hierarchical representations of knowledge in Medical Informatics. In: *ITCH 2007*, Victoria, BC (2007)
6. Kanerva, P.: *Sparse distributed memory*. MIT Press, Cambridge (1988); xxii, 155
7. Ellard, D., Ellard, P.: *S-Q Course Book* (2003), <http://www.eecs.harvard.edu/~ellard/Courses/> (cited January 24, 2006)
8. Landauer, T., Dumais, S.: A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. *Psychological Review* 104(2), 211–240 (1997)
9. Wille, R.: Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies. In: Ganter, B., Stumme, G., Wille, R. (eds.) *Formal Concept Analysis*. LNCS, vol. 3626, pp. 1–33. Springer, Heidelberg (2005)
10. Erdos, P.L., Sziklai, P., Torney, D.C.: A finite word poset. *The Electronic Journal of Combinatorics* 8(2), 1–10 (2001)
11. Ventos, V., Soldano, H.: Alpha Galois Lattices: An Overview. In: Ganter, B., Godin, R. (eds.) *ICFCA 2005*. LNCS, vol. 3403, pp. 299–314. Springer, Heidelberg (2005)
12. Díaz-Agudo, B., González Calero, P.A.: Classification based retrieval using formal concept analysis. In: Aha, D.W., Watson, I. (eds.) *ICCBR 2001*. LNCS, vol. 2080, p. 173. Springer, Heidelberg (2001)
13. Bichindaritz, I.: Memory Structures and Organization in Case-based Reasoning. In: Perner, P. (ed.) *Case-based Reasoning on Images and Signals*, pp. 175–194. Springer, Heidelberg (2008)
14. Kohonen, T.: *Self-Organizing Maps*. In: Huang, T.S. (ed.), 3rd edn. Springer series in information sciences, vol. 30, p. 501. Springer, Heidelberg (2001)
15. Frick, A., Ludwig, A., Mehldau, H.: A Fast Adaptive Layout Algorithm for Undirected Graphs. In: *DIMACS Workshop on Graph Drawing*. Springer, Heidelberg (1995)
16. Knuth, D.E.: *The art of computer programming: Sorting and Searching*, 2nd edn., vol. 3. Addison-Wesley, Reading (1997)
17. Manning, C.D., Schütze, H.: *Foundations of Statistical Natural Language Processing*, xxxvii, p. 680. MIT Press, Cambridge (1999)
18. Bichindaritz, I.: Memory Organization As the Missing Link Between Case Based Reasoning and Information Retrieval in Biomedicine. In: *ICCBR 2005 Workshop on CBR in the Health Sciences* (2005)
19. Hersh, W.R., Hickam, D.H., Haynes, B.: A performance and failure analysis with a MEDLINE test collection. *J. Am. Med. Inform. Assoc.* 1, 51–60 (1994)

# Robust Measures of Complexity in TCBR

M.A. Raghunandan, Sutanu Chakraborti, and Deepak Khemani

Department of Computer Science and Engineering,  
Indian Institute of Technology Madras,  
Chennai-600036, India

maraghu@cse.iitm.ac.in, {sutanuc,khemani}@iitm.ac.in

**Abstract.** In TCBR, complexity refers to the extent to which similar problems have similar solutions. Casebase complexity measures proposed are based on the premise that a casebase is simple if similar problems have similar solutions. We observe, however, that such measures are vulnerable to choice of solution side representations, and hence may not be meaningful unless similarities between solution components of cases are shown to corroborate with human judgements. In this paper, we redefine the goal of complexity measurements and explore issues in estimating solution side similarities. A second limitation of earlier approaches is that they critically rely on the choice of one or more parameters. We present two parameter-free complexity measures, and propose a visualization scheme for casebase maintenance. Evaluation over diverse textual casebases show their superiority over earlier measures.

## 1 Introduction

Textual Case Based Reasoning (TCBR) involves answering queries expressed in free-form text, by reusing past problem solving episodes, which are themselves recorded as cases expressed in text. A case in TCBR is typically composed of a problem and its associated solution, both of which are unstructured texts. The underlying assumption is that a case whose problem description is sufficiently “similar” to a query, is likely to have a solution which is relevant in answering the query. Computing similarity between textual problem descriptions of cases is non-trivial, since surface level features like words may not have direct bearing with the underlying meaning. Thus, free-form descriptions are often mapped to richer representations, to ensure that the comparisons are meaningful; this may involve using a combination of background (domain specific) and linguistic knowledge. A central problem in TCBR is choosing the right representation for the problem descriptions. The proof of the pudding is in the eating, and thus a good representation is one that maximizes the chance that a retrieved case will be useful in solving the query problem. Evaluation measures from Information Retrieval like accuracy, precision and recall have been used in TCBR to measure retrieval effectiveness and can act as posterior estimates of how good a representation is. Recent studies have attempted to characterize a casebase using complexity measures like cohesion [8], alignment [9], and global alignment

[13], and to demonstrate that knowledge of these characteristics can be useful in predicting the CBR system performance even in the absence of unseen queries and corresponding human relevance judgements.

Complexity measures find several applications. First, we can use complexity to guide our choice of casebase representations, with the goal of improving retrieval effectiveness. Secondly, given a choice of representation, we can compare across different casebases and use these measures to explain varying retrieval effectiveness. Thirdly, problematic cases or features can be identified, and this may be useful in suggesting case-base maintenance operations (like deleting or modifying existing cases, or adding new cases and/or features) that can lead to improved competence of the system.

This paper revisits the assumptions behind the measures proposed so far. We argue that these measures are brittle in two ways. Firstly, these measures are founded on the following hypothesis:

*A case-base is simple if solutions of similar problems are similar.*

Raghunandan et al. [13] evaluate the goodness of complexity measures by showing that they positively correlate with accuracies over diverse casebases. Chakraborti et al. [11] use a similar approach, but across different representations of the same casebase. In this paper, we highlight a salient limitation of these approaches, in that the measures are critically dependent on the choice of solution side representations. Thus, we could obtain a high correlation simply by arbitrarily tweaking solution representations such that the resulting solution similarities result in complexity measures that show strong positive correlation with accuracy. Note that a given choice of problem representations can thus give rise to very different complexity measures, thus rendering comparisons across representations and across casebases meaningless. We argue that the governing criterion should be changed to the following:

*A case base is simple if small changes in the query only effect small changes in the “result set”, i.e. the set of cases regarded as relevant by an expert.*

*Alternatively, A casebase is simple if result sets of similar queries are similar.*

In effect, we assert that for complexity measures proposed in [13] to work, we need to ensure that solution side similarities are supported by human judgements. This raises the question: if both accuracy and complexity measures are reliant on human judgements, why would we need complexity measures in the first place? We argue that evaluation measures like accuracy and complexity measures have complementary goals. While accuracy is a black box estimate and is agnostic to the technique used internally to perform retrieval, complexity is strongly tied to instance based learning: complexity tells us how well suited CBR (the instance based learner) is, for the problem at hand.

A second limitation of the complexity measures is that all of them involve more than one free parameter, which need to be set appropriately, often using trial and error. The correlation studies reported are thus critically dependent on these parameter settings. In this paper we present two parameter free evaluation measures that overcome these limitations. In addition to these measures we also present a novel visualization technique inspired by one of these complexity

measures. The visualization technique allows us to identify problematic pairs of cases, and would thus be useful for maintenance tasks.

The paper is organized as follows. Section 2 surveys related work and positions our work in context. Section 3 argues that solution similarity need to corroborate with human judgements, for complexity measures to be meaningful. We provide empirical results to substantiate this. Section 4 presents two robust algorithms for measuring complexity, and also proposes a novel visualization technique for casebase maintenance. Section 5 discusses our evaluation methodology and presents empirical findings on how well the two measures fare over datasets of varying size and complexity. We highlight our main contributions and conclude in Section 6.

## 2 Related Work

Researchers in both TCBR and Information Retrieval (IR) have been interested in the problem of characterizing a collection of documents using complexity measures that can be predictive of the retrieval effectiveness. In section 2.1, we establish a mapping between CBR and IR retrieval goals. We then take a quick tour of existing work, identify their shortcomings and motivate our current work in that context.

### 2.1 TCBR and IR

IR aims at retrieving documents from a collection that are relevant to answering a query. In TCBR the query is a problem description, and the task is to suggest a solution based on cases whose problem descriptions are similar to the query. While past studies have highlighted differences between CBR and IR [3], we propose that each document in IR can, in fact, be treated as a case. Though the goal is to retrieve a documents, the documents are not compared directly with the query. Rather, a characterization of these documents in terms of a set of underlying indices is compared against the query and relevance rankings produced. We can draw parallels between the document characterizations in IR and problem representations of textual cases in TCBR, in that they are both matched against the query during retrieval. In TCBR, a case base is simple if solutions corresponding to similar problem descriptions are regarded as similar by humans. Extending this to IR, a document collection in IR is regarded as simple if we can ascertain that documents with similar characterizations will be adjudged as similar by humans. Thus, complexity measures designed for TCBR are likely to be relevant for IR as well.

### 2.2 Complexity Measures in TCBR

Raghunandan et. al. [13] present a consolidated study, and empirical comparison, of three different complexity measures proposed in the context of TCBR. Two of them, namely case cohesion and case alignment are local measures in the



sense that they assign an alignment score of a particular case based on its neighborhood, and then aggregate the alignments of all the cases to obtain the casebase alignment. The third was a global measure, which compared the linear ordering of cases on the problem and the solution side, to arrive at a complexity estimate.

The case cohesion measure presented in Lamontagne [8] defines alignment in terms of the degree of overlap between the nearest neighbours of a case on the problem and the solution side. The case-specific cohesion values can be aggregated to define a cohesion value for the entire casebase. One limitation of the approach is that the problem and solution sides are treated symmetrically. This goes against the intuition that similar problems having dissimilar solutions harm alignment a lot more than dissimilar problems having similar solutions. One more disadvantage of this measure is that all neighbouring cases are treated equally, irrespective of their distance to the case in question. The case alignment measure presented in [9] overcomes some limitations of cohesion. Firstly, it restricts attention to cases in the neighbourhood of a given case based on problem side similarities, thus respecting asymmetry between problem and solution side similarities. Secondly, the effect of each case on the alignment is weighted by its problem similarity to the target case. Hence nearby cases have a more pronounced effect on the measure than cases which are farther away. The global alignment measure presented in [13] is based on the notion of stacking, which has the effect of generating an ordered list of cases such that cases which are similar to each other are stacked closed to each other. Two such lists are generated, one using problem side similarities, and the the other using solution side similarities. The key intuition is that well aligned casebases should lead to good solution side clustering of cases when the problem side ordering is imposed on the solution side. As discussed in Section 4, imposing a linear order on cases may lead to a representation that is not rich enough to capture associations between cases.

We note that all work till date define alignment measures in terms of solution side similarities derived from solution representations of cases. As explained in Section 3, this does not always work. In addition to this, all the methods studied earlier are (sometimes critically) dependent on the setting of some parameters. There are no standard values of these parameters that have been found to work in a large variety of setting, nor are there any guidelines for choosing their values. Hence it becomes difficult to determine which value of the parameter gives an accurate alignment value.

### 3 Challenges in Estimating Solution Similarity

#### 3.1 Pitfalls in Earlier Approaches

A case-base is said to be simple or well aligned when similar problems have similar solutions. But when can we say that two solutions are similar? Earlier approaches have taken some measure of text similarity on the problem and solution side, and used that to calculate alignment. This alignment measure is

then used to guide design choices such as representation and similarity measure chosen. We observe however, that the solution representation can be arbitrarily chosen such that the solution similarities lead to high alignment even though this is barely indicative of system performance in the face of human evaluation. Therefore we must be careful in estimating solution similarities. In classification problems, where class labels are used as solutions, when two cases have the same class label, then they are similar, and if they have different class labels, then they are dissimilar. Further, if the class labels were to come from a taxonomy or an ordering then one could define a similarity measure on the class labels. In both these cases, the solution similarities are implicitly derived out of human judgments. Hence higher alignment can be expected to predict higher classification accuracies. However, in unsupervised settings, where the solutions are in free-text form, estimation of solution similarities is not straightforward. Earlier works ([8], [9], [13]) use representations of solution texts to estimate similarities; we demonstrate in section 3.2 why this is not a good idea. To overcome this shortcoming, we need indirect ways of estimating solution similarities from human relevance judgements; one such approach is presented in section 3.3.

### 3.2 Empirical Demonstration

We consider the NHS medical dataset [13], which consists of over 4000 incident reports in the medical domain. The problem describing an incident and solution describing an action taken are in free text form. In addition, there are six labels for every case, that describe different aspects of the case. We generated eleven different case-bases from the original case-base. Dataset 1 is the least complex, having just 1 class, whereas dataset 11 is the most complex with 11 classes. Classification accuracy was calculated with each of the labels as the class label. Alignment was measured, first using the solution text, and then using the labels. The results are shown in table 1. We observe that in most cases, the correlation between alignment and accuracy was better when a combination of labels were used, than with solution text. This shows that the labels, which are a form of user feedback, are more reliable indicators of solution similarity than free text.

**Table 1.** Alignment - Medical dataset

Class Label	Corr. with text	Corr. with labels
care stage	-0.598	0.050
detail	-0.629	0.202
adv_event	-0.516	0.280
result	0.406	0.235
severity	0.138	-0.248
type	-0.291	0.102

That solution similarities based on solution text alone may not lead to robust estimates can also be seen from results reported by Gabrilovich et. al. [12] who carry out experiments over a collection of 50 documents from the Australian Broadcasting Corporations news mail service, which were paired in all possible

ways and 8-12 human judgements were averaged for each pair. A bag of words similarity measure had a poor correlation (0.1 - 0.5) with human judgements, whereas LSA based similarities yielded a correlation of 0.6.

### 3.3 Estimating Similarities in the Presence of Human Judgment

The similarity between two problems is determined by the representation and similarity measures used for retrieval. However, estimating the similarity between two solutions is not so straightforward. This is because a solution does not participate in retrieval, hence we have no principled way of choosing between representations. To circumvent this problem, we observe that we can have a preference ordering of representations, based on how well the solution similarities generated therefrom correspond with human judgements.

For text data, similarity is calculated by using a cosine measure on a vector representation of the text in the term space, or in some derived space, like the LSA concept space described in Deerwester et. al. [11], or the LDA topic space described in Blei et. al. [7]. In an IR setting, given a set of documents  $D$ , and a query  $q_i$ , a human can identify the set of documents  $rel_i \subseteq D$  that are relevant to the query. The set of documents  $irrel_i = D - rel_i$  are assumed irrelevant. In the TCBR context, we can treat the problem components of cases as queries, and the lists of relevant cases  $rel_i$  (cases whose solutions are deemed by an expert to be relevant in answering the query) as the solutions  $s_i$ . We can now use the overlap between two relevance lists as an estimate of the similarity between the two solutions. Thus,

$$sim(s_i, s_j) \sim sim(rel_i, rel_j) = \frac{|rel_i \cap rel_j|}{|rel_i \cup rel_j|}$$

An interesting research direction is to look at the problem of solution side similarity estimation, when very little explicit human judgement is available. Researchers in IR have looked at the problem of implicit relevance feedback [4], which is relevant in this context.

## 4 Calculating Complexity

Let us consider two out of the four possible relationships between two cases in the case-base. If two cases are dissimilar on the problem side, and yet have similar solutions, this may not adversely affect complexity, but we have an interesting scenario that warrants a closer look. However, two cases which are similar on the problem side having dissimilar solutions contribute to increased complexity in the case-base. Neighbourhoods of such cases may pose a challenge for instance based learners, since retrieval will not be robust to very small changes in the query. Complexity measures are designed to quantify and aggregate the contributions of such pathological pairs of cases.

In this section we look at two measures of calculating alignment between the problem and solution spaces. The first measure calculates a spanning tree on the problem and solution side, and then compares them to determine the alignment. The second is based on the correlation between the problem and solution similarity values. These measures address the limitations of the previous measures, as identified in section 2. We also describe a visualization of the case-base that naturally follows from the spanning tree approach.

#### 4.1 MST Method

The minimum spanning tree of a graph, is a tree, the sum of whose edges weights is minimal. Consider a fully connected, undirected graph with the nodes representing cases, and edge weights representing the distance between cases. We propose that the minimal spanning tree (MST) of this graph captures, in essence, the most important links in the case-base. We have two such graphs, one each on the problem and solution side; the corresponding MSTs can be compared to give us an estimate of complexity. A simple casebase will have many edges in common between the two trees, whereas a complex casebase will have largely different trees.

Let us consider a simple case-base that has five cases. The problem- and solution-side graphs are as shown in figures 1a and 1b. (To avoid clutter, all edges are not shown.) The edges which belong to the MST are shown dark, whereas others are dotted. Since the two trees are different, we can see that the case-base is complex. To calculate alignment, we assign the solution-side weights to the problem-side tree, to get a sum of weights,  $cost$ . The sum of weights of the solution side MST,  $cost_{min}$ , is the minimum sum for any tree on the solution-side. We can similarly compute a  $cost_{max}$ , which is the sum of weights of the maximal spanning tree on the solution side. We now define the alignment between the problem and solution space as,

$$alignMST = \frac{cost - cost_{min}}{cost_{max} - cost_{min}}$$

#### 4.2 Correlation Method

In well-aligned case bases, cases having a high similarity on the problem side are expected to have a high similarity on the solution side as well. This observation leads to the consideration of the correlation between the problem similarities, and the solution similarities of cases, as a measure of alignment. Directly taking the correlation between these two sets of values doesn't bring out the asymmetry between the problem and solution side. Essentially, case pairs which have higher problem similarity should have a greater bearing on the alignment. This leads us to use weighted correlation measure between the two sets of similarities, with the weights equal to the problem similarities. This formulation is given below:

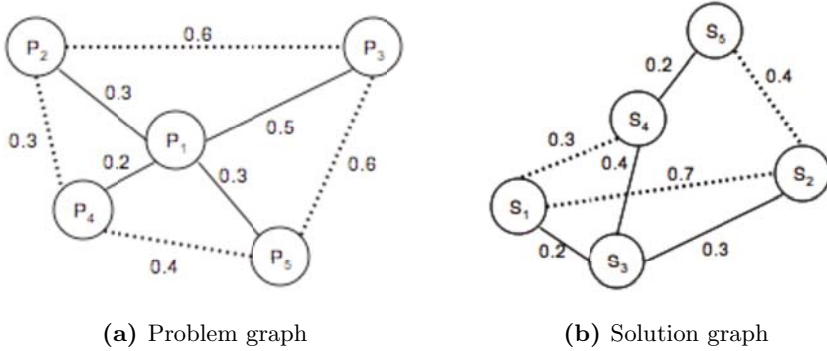


Fig. 1. Case-base graphs

In a case-base having  $n$  cases, consider the set of problem similarity values given by,

$$PS = \{ps_{i,j} : 1 \leq i < j \leq n\}$$

where  $ps_{i,j}$  is the similarity of the problems of cases  $i$  and  $j$ . The set of solution similarities  $SS$  is similarly defined. The alignment between the problem and solution space is given as,

$$alignCorr = wtCorr(PS, SS, PS)$$

where,

- the weighted correlation function  $Wt\_Corr$  between two sets of values  $x$  and  $y$ , with weights  $w$ , is given by,

$$wtCorr(x, y, w) = \frac{wtCov(x, y, w)}{wtCov(x, x, w) * wtCov(y, y, w)}$$

- the weighted covariance function  $Wt\_Cov$  between two sets of values  $x$  and  $w$ , with weights  $w$ , is given by,

$$wtCov(x, y, w) = \frac{\sum_i w_i(x_i - m(x, w))(y_i - m(y, w))}{\sum_i w_i}$$

- and the weighted mean  $m(x, w)$  of a set of values  $x$ , with weights  $w$ , is given by,

$$m(x, w) = \frac{\sum_i w_i x_i}{\sum_i w_i}$$

For example, consider the set of problem and solution similarities,

$$PS = \{0.9, 0.7, 0.6, 0.5, 0.4, 0.3\}, SS = \{0.6, 0.7, 0.6, 0.5, 0.4, 0.3\}$$

The correlation between these two sets yields the value 0.539, but the weighted correlation gives 0.450, thus greatly penalizing similar problems having dissimilar solutions.

### 4.3 Visualization Using MST

Visualization is aimed at gaining a bird's eye view of a system usually for the purpose of design, tuning and maintenance. Visualization of a case base should ideally display the structure and properties of the cases highlighting properties like alignment, but also bringing out deficiencies in coverage and regions of misalignment. A good visualization should be powerful enough to capture and present the overall structure of a case-base in a single picture, and at the same time sensitive enough to identify anomalies and interesting patterns in small regions of the case-base. The method of visualization used will depend on the objective of visualization. We focus on complexity-driven visualization. The visual representation of a case should give a clear idea of the degree of alignment of the case-base as a whole, as well as those of the various regions of the case-base.

Chakraborti et. al. [11] give a method of visualization called stacking, based on the representation of a case-base as a document-feature matrix. This method helps in identifying interesting patterns in the arrangement of cases and features. However, it imposes a linear ordering on the cases, and doesn't capture the relationships between the problem and the solution side. Our proposed method addresses these two limitations of the stacking approach.

The MSTs of the problem and solution sides of a case-base (section 4.1) give a good idea of the structure of the case-base. It would be useful if we could combine the information in both the graphs into a single graph. To this effect, the case-base is represented as a graph, with the nodes representing the cases. Each edge  $e_{ij}$  between two cases  $i$  and  $j$ , having solution similarity  $ss_{ij}$ , solution distance  $sd_{ij} = 1 - ss_{ij}$ , and problem similarity  $ps_{ij}$  has these characteristics:

- The edge belongs to the MST of the solution graph.
- The length of the edge is directly proportional to the solution distance between the two cases.

$$length(e_{ij}) \propto sd(i, j)$$

Hence, edges between cases which are similar on the solution side will be shorter.

- The thickness of the edge is directly proportional to the problem similarity between the two cases.

$$thickness(e_{ij}) \propto ps(i, j)$$

Hence, edges between cases which are similar on the problem side will be thicker. Thus ideally as an edge becomes longer it should also become thinner. Thick, long edges, point to similar problems having dissimilar solutions, which indicates high complexity.

- In addition to the above, we employ a color encoding scheme to depict which side of the ideal an edge deviates on (hue), and by how much (saturation), according to the following equations:

- $hue = BLUE$ , if  $ps_{ij} \geq ss_{ij}$ , and  $RED$ , if  $ss_{ij} > ps_{ij}$
- $saturation = 1 - \frac{ss_{ij}}{ps_{ij}}$ , if  $ps_{ij} \geq ss_{ij}$ , and  $1 - \frac{ps_{ij}}{ss_{ij}}$ , if  $ss_{ij} > ps_{ij}$

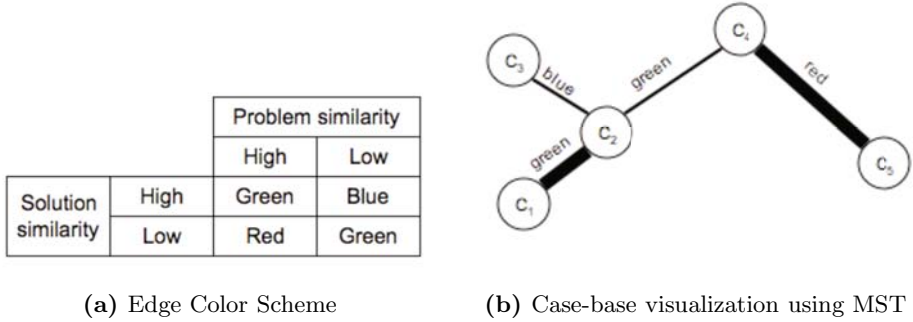


Fig. 2. Visualization using MST

This roughly translates to the color scheme shown in figure 2a. An example visualization is shown in figure 2b.

From this figure, we can clearly make out the similarity of the cases on the problem and the solution side. The coloring scheme helps us to quickly narrow down on interesting and complex regions of the case-base, thus aiding in maintenance tasks such as addition and deletion of cases. For example, the red (thick and long) edges (C4C5, in this case), show the regions where solution side similarity is lower than that of the problem side, indicating a misalignment. We can choose to add/delete/modify some of the cases in such regions, to improve the alignment.

For the purpose of experimentation, we created two synthetic datasets. In the first one, called Syn-sim (table 2), similar problems have similar solutions, whereas in the second, called Sym-dissim (table 3), similar problems have dissimilar solutions.

Table 2. Some cases from synthetic dataset 1 - Syn-sim

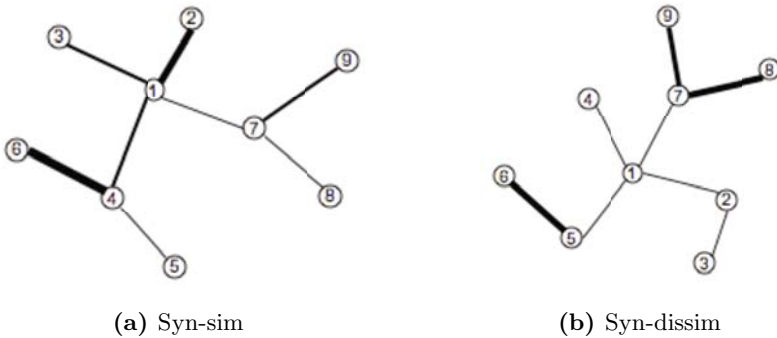
Case No.	Problem	Solution
1	Translation of instructions written in a high level language, to assembly code.	computer, programming, compiler
2	Description of a set of instructions, which have to then be coded as a program.	computer, programming, algorithm
7	Increasing number of cars, bikes, buses, and autos, competing with each other for space.	traffic, congestion, vehicle
8	Jams on the roads, incorrect signalling, and no footpaths. This means hell for people who like to walk.	traffic, congestion, pedestrian

The figure 3 shows the visualization of the synthetic datasets by the above method.

This method can be extended to be more dynamic. The visualization can be hierarchical, wherein many cases are collapsed into one node in the graph. The node can be colored to show misalignment among its cases. The user can then decide to zoom in on that node, to greater level of detail.

**Table 3.** Some cases from synthetic dataset 2 - Syn-dissim

Case No.	Problem	Solution
1	The program can be easily selected by using either the mouse or the keyboard.	computer, interface, user-friendly
2	The mouse delighted us by playing the keyboard like a maestro. Everyone enjoyed the program.	music, audience, performance
8	All was lost when the king was captured in the end by the enemy and killed.	war, death, defeat
9	To protect the king from being captured, good players move the king only towards the end, after many pieces are off the board.	chess, strategy, game

**Fig. 3.** Synthetic dataset visualization

## 5 Evaluation

TCBR datasets, especially those with human relevance judgments, are hard to come by. For our experiments, we required datasets from which we can indirectly obtain human judgments in solution similarities. We used four datasets in our experiments. For each of the datasets, classification accuracy (acc) was calculated using weighted k-NN. The alignment was calculated using each of the measures - weighted correlation (alignCorr, section 4.2), MST (alignMST, section 4.1), flips (alignFlips, [11]), and global alignment (alignGame, [13]). In addition, experiments were conducted using two local measures of alignment, the case alignment (alignMassie, [9]) and the case cohesion (alignCohesion, [8]). The measures, alignCorr, and alignMST are parameter free, hence only one alignment value is reported for each dataset. However, the alignFlips, alignGame, alignMassie, and alignCohesion measures require the neighborhood size parameter to be set. For these measures, two values of alignment, corresponding to different parameter values, are reported.

### 5.1 Synthetic Datasets

Consider the synthetic dataset in 4.3. We would expect a good alignment measure to assign a high alignment to the Syn-sim, and a low alignment to Syn-dissim.



**Table 4.** Alignment - Synthetic dataset

Dataset	acc	alignCorr	alignMST	alignFlips	alignGame
Syn-sim	0.278	0.571	0.866	0.900,0.900	0.697,0.737
Syn-dissim	0.211	0.497	0.500	0.919,0.919	0.596,0.596

The alignment results for these datasets, using the measures described are as shown in table 4.

The alignCorr, alignMST and alignGame values agree with the accuracy results, whereas alignFlips does not, and in fact indicates the opposite. The local measures alignCohesion and alignMassie perform poorly as well.

## 5.2 Deerwester Dataset

The Deerwester toy dataset (table 5) is a collection of nine documents, five of which are about human-computer interaction, and four about graphs. The vocabulary used in the cases is aligned well with the class label. Hence the classification accuracy is high. With the addition of some random terms to each of the cases, however, the accuracy decreases. The alignment results are as in table 6. DW-N0 is the original dataset DW-Nx has ‘x’ noise terms added to the problem part of each case.

We see that the alignCorr, alignMST, and alignFlips have high correlation with accuracy, although the alignFlips values are not indicative of the complexity. alignGame values do not agree with the complexity values. The local measures alignCohesion and alignMassie perform well, with correlations in the range [0.704,0.976] and [0.799,0.870] respectively, although the value depends on the choice of parameter.

**Table 5.** Deerwester dataset

Problem	Solution
Human machine interface for Lab ABC computer applications	hci
A survey of user opinion of computer system response time	hci
The EPS user interface management system	hci
System and human system engineering testing of EPS	hci
Relation of user perceived response time to error measurement	hci
The generation of random, binary, unordered trees	graphs
The intersection graphs of paths in trees	graphs
Graph minors 4: widths of trees and well quasi ordering	graphs
Graph minors: A survey	graphs

**Table 6.** Alignment - Deerwester dataset

Dataset	acc	alignCorr	alignMST	alignFlips	alignGame
DW-N0	0.889	0.683	1.000	0.500,0.500	0.421,0.619
DW-N1	0.778	0.632	0.857	0.376,0.376	0.737,0.774
DW-N2	0.667	0.589	0.714	0.376,0.376	0.737,0.774
DW-N3	0.667	0.510	0.429	0.376,0.376	0.737,0.774
Correlation	0.894	0.866	0.870,0.870	-0.870,-0.870	

### 5.3 NHS Medical Dataset

The NHS dataset described in section 3.2 has a “care stage” label which categories the case into one of 18 categories. Using weighted 3-nearest neighbour on the problem text, the classification accuracy was calculated with the “care stage” attribute as the class label. We found that the classification accuracy was poorer in datasets having more class labels. A study of alignment was carried out in [13], using solution text similarities. These may not be indicative of human judgments. To overcome this problem we considered the class labels as the solutions, and, as pointed out in 3.2, the alignment values are more indicative of accuracy. The alignment values, using a variety of techniques, as well as their correlation with accuracy, are shown in table 7.

**Table 7.** Alignment - Medical dataset

Dataset	acc	alignCorr	alignMST	alignFlips	alignGame
1	1.000	0.691	0.863	0.283,0.303	0.757,0.778
2	0.830	0.785	0.703	0.455,0.374	0.635,0.728
3	0.758	0.826	0.692	0.357,0.316	0.750,0.832
4	0.690	0.849	0.593	0.485,0.465	0.672,0.778
5	0.600	0.795	0.553	0.414,0.394	0.764,0.839
6	0.539	0.816	0.548	0.396,0.376	0.799,0.864
7	0.469	0.757	0.532	0.381,0.351	0.828,0.895
8	0.423	0.722	0.450	0.447,0.417	0.814,0.887
9	0.455	0.688	0.413	0.469,0.469	0.829,0.851
10	0.400	0.647	0.355	0.495,0.475	0.782,0.841
11	0.343	0.616	0.310	0.541,0.510	0.762,0.832
Correlation		0.431	0.970	-0.679,-0.711	-0.592,-0.726

The alignCorr has low correlation with accuracy, while alignMST does very well. Both of these outperform the alignFlips and alignGame measures, which show a negative correlation. Among local measures, alignCohesion gives correlation values in the range [-0.927,0.547], showing high sensitivity to the choice of the neighborhood size, while alignMassie is more stable, giving correlation between 0.978 and 0.987.

### 5.4 20 Newsgroups Dataset

The 20 Newsgroups dataset [2] contains data from 20 different newsgroups. The partitions of the data are derived from a tree of topics. High level topics include “comp”, “talk”, “sci”, “rec”, etc., and these are further divided into subgroups such as “rec.sport.baseball”, “talk.politics.guns”, etc. We created 5 datasets of varying complexity from the original dataset. The alignment results for these using the various measures, are given in table 8.

We observe that although alignCorr and alignMST have high correlation with accuracy, the alignCorr values do not show the range of variation in accuracy. alignMST is better in this regard. alignFlips shows negative correlation with accuracy, while alignGame is not able to differentiate the complexity of the

**Table 8.** Alignment - 20 Newsgroups dataset

Dataset	acc	alignCorr	alignMST	alignFlips	alignGame
1	0.740	0.536	0.755	0.045,0.023	0.976,1.000
2	0.656	0.529	0.643	0.053,0.030	0.976,1.000
3	0.630	0.523	0.602	0.061,0.037	0.975,1.000
4	0.578	0.519	0.598	0.074,0.040	0.966,1.000
5	0.501	0.519	0.512	0.050,0.050	1.000,1.000
Correlation	0.929	0.973	-0.360,-0.988	-0.531,-	

datasets. Among local measures, again alignCohesion shows high sensitivity to parameter setting, with correlation varying from -0.885 to 0.955. alignMassie is more stable, with correlations in the range [0.949,0.925].

## 6 Conclusions

The main contributions of this paper are threefold. Firstly, we highlight that existing complexity measures can provide meaningful predictions about system performance only when solution similarities are directly or indirectly supported by human judgements. In the light of this observation, we redefine the problem of complexity measurement and discuss an approach for estimating solution similarity based on human relevance judgements. Secondly, we propose two robust complexity measures that, unlike previously proposed measures, need no parameter settings. Thirdly, we present a visualization scheme that aids maintenance tasks by identifying pairs of cases that contribute to increasing complexity. In future, we plan to incorporate more features into the current visualization tool, and work on devising interesting ways for estimating solution similarities based on scant human feedback. In practical settings, indirect approaches to infer relevance judgements as in IR, such as click behaviour, could be used; with more feedback accumulating, solution similarities are expected to stabilize with time. As a community, we need to look at interesting ways of collaborating to create TCBR datasets with human judgements, so that retrieval approaches as well as complexity measures can be standardized.

## References

1. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* 41(6), 391–407 (1990)
2. Mitchell, T.: *Machine Learning*. Mc Graw Hill International (1997)
3. Lenz, M., Ashley, K.: *Papers from the AAAI Workshop*. AAAI Press, Menlo Park (1998)
4. Kelly, D., Belkin, N.J.: Reading Time, Scrolling, and Interaction: Exploring Implicit Sources of User Preferences for Relevance Feedback During Interactive Information Retrieval. In: *Proc. of the SIGIR* (2001)
5. Bergmann, R.: *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Springer, Heidelberg (2002)

6. Singh, S.: Prism, Cells and Hypercuboids. *Pattern Analysis and Applications* 5 (2002)
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3(2003), 993–1022 (2003)
8. Lamontagne, L.: Textual CBR Authoring using Case Cohesion. In: *TCBR 2006 - Reasoning with Text, Proceedings of the ECCBR 2006 Workshops*, pp. 33–43 (2006)
9. Massie, S., Craw, S., Wiratunga, N.: Complexity profiling for informed case-base editing. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 325–339. Springer, Heidelberg (2006)
10. Vinay, V., Cox, J., Milic-Fralyng, N., Wood, K.: Measuring the Complexity of a Collection of Documents. In: Lalmas, M., MacFarlane, A., Rüger, S.M., Tombros, A., Tsirikika, T., Yavlinsky, A. (eds.) *ECIR 2006*. LNCS, vol. 3936, pp. 107–118. Springer, Heidelberg (2006)
11. Chakraborti, S., Beresi, U., Wiratunga, N., Massie, S., Lothian, R., Watt, S.: A Simple Approach towards Visualizing and Evaluating Complexity of Textual Case Bases. In: *Proc. of the ICCBR 2007 Workshops* (2007)
12. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: *Proceedings of The Twentieth International Joint Conference for Artificial Intelligence, Hyderabad, India*, pp. 1606–1611 (2007)
13. Raghunandan, M.A., Wiratunga, N., Chakraborti, S., Massie, S., Khemani, D.: Evaluation Measures for TCBR Systems. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) *ECCBR 2008*. LNCS, vol. 5239, pp. 444–458. Springer, Heidelberg (2008)
14. Correlation - Wikipedia, <http://en.wikipedia.org/wiki/Correlation>

# S-Learning: A Model-Free, Case-Based Algorithm for Robot Learning and Control

Brandon Rohrer

Sandia National Laboratories, Albuquerque, NM 87185, USA  
brrohre@sandia.gov

**Abstract.** A model-free, case-based learning and control algorithm called S-learning is described as implemented in a simulation of a light-seeking mobile robot. S-learning demonstrated learning of robotic and environmental structure sufficient to allow it to achieve its goal (reaching a light source). No modeling information about the task or calibration information about the robot's actuators and sensors were used in S-learning's planning. The ability of S-learning to make movement plans was completely dependent on experience it gained as it explored. Initially it had no experience and was forced to wander randomly. With increasing exposure to the task, S-learning achieved its goal with more nearly optimal paths. The fact that this approach is model-free and case-based implies that it may be applied to many other systems, perhaps even to systems of much greater complexity.

## 1 Introduction

S-learning is a general learning and control algorithm modeled on the human neuro-motor system [2,8,9]. It is model-free in the sense that it makes no assumptions about the structure or nature of the system being controlled or its environment. S-learning accomplishes this using case-based reasoning. It uses previous experience to help it select actions. This paper describes the implementation of S-Learning in computer code and the application of S-learning to a simulated mobile robot.

### 1.1 Relation to Previous Work

Most approaches to robot control assume the existence of an explicit system model. Even the majority of learning algorithms take the form of a search in parameter space, with the underlying structure determined beforehand. Other methods make a less constraining assumption: that the vectors of state information occupy a metric space. This allows the use of distance metrics, such as the  $L^2$  norm, to interpret its state history. These include finite state machines [11], variants of differential dynamic programming [7,12], the Parti-game algorithm [6], and probabilistic roadmaps [3]. But even this seemingly benign

assumption implies a good deal about the system being modeled. It is violated by any sufficiently non-smooth system, such as one containing hard-nonlinearities or producing categorical state information.

There are still a number of algorithms that are similar to S-learning in that they make no assumptions about the system being learned and controlled. These include Q-learning [13], the Dyna architecture [10], Associative Memory [4], and neural-network-based techniques including Brain-Based Devices [5] and CMAC [1]. These approaches, together with S-learning, can be categorized as reinforcement learning (RL) algorithms, or solutions to RL problems. However, these all assume a static reward function, where S-learning does not.

## 1.2 Dynamic Reinforcement Learning Problem Statement

To be more precise, S-learning addresses a general class of reinforcement learning (RL) problem, referred to hereafter as the dynamic RL problem: how to maximize reward in an unmodeled environment with time-varying goals. More specifically, given discrete-valued action (input) and state (output) vectors,  $a \in \mathcal{A}$  and  $s \in \mathcal{S}$ , and an unknown discrete-time function  $f$ , such that

$$s_t = f(a_{i \leq t}, s_{i < t}, t), \quad (1)$$

(where the notation  $a_{i \leq t}$  denotes the set of all  $a_i$  such that  $i \leq t$ ) and a scalar reward,  $r$ , and known reward function,  $g$ , such that

$$r_t = g(s_{i \leq t}, t), \quad (2)$$

maximize the total reward over time:

$$V = \sum_{i=0}^{\infty} r_i \quad (3)$$

Equation 3 shows an infinite-horizon formulation, but finite- and receding-horizon variations of the dynamic RL problem are similarly structured.

The dynamic RL formulation is relevant to a large class of problems. It is applicable in instances where 1) the model is unavailable and 2) the reward function varies with time. Models may be unavailable for a number of reasons. Systems may be too complex to model accurately with the resources available. Also, systems may have characteristics that vary with age, such as joint friction or tire pressure, or may even have minor sensor and actuator failures. Time varying reward functions are introduced whenever the system's goals are modified, as in response to an operator command. Despite the importance of the dynamic RL problem, no other published solutions exist. The nature of the dynamic RL problem—that the only information available is the robot's action-state history—suits it well to a case-based reasoning approach.

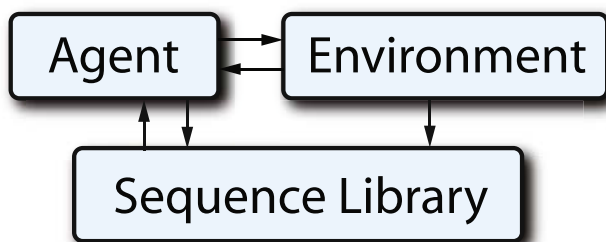
## 2 Method

S-learning operates by recording sequences of state-action pairs. The resulting libraries contain a reduced version of the system's history, a system memory. The memory can then be used to make predictions and guide the selection of the system's actions. When the system encounters a previously-experienced state, it retrieves sequences beginning with that state. The system can then re-execute the actions of recalled sequences that terminate in a reward state, as in a case-based approach.

### 2.1 S-Learning Algorithm

S-learning handles state-action ( $s$ - $a$ ) pairs,  $\sigma$ . An ordered sequence of state-action pairs is called a *sequence*,  $\phi$ , and an unordered collection of  $\phi$  is a *library*,  $\kappa$ . Both  $\phi$  and  $\kappa$  may have any length of one or more, given by  $n_\sigma$  (number of state-action pairs) and  $n_\phi$  (number of sequences), respectively.

An S-learning implementation can be broken into three main function blocks: the Agent, the Environment, and the Sequence Library. (Figure [1](#))



**Fig. 1.** Block diagram of S-learning. The Environment represents the system dynamics,  $f$ , and the Agent contains the reward function,  $g$ . The Sequence Library is created from the time history of  $s$ - $a$  pairs.

**Environment.** The Environment is the embodiment of the system dynamics,  $f$  (Equation [1](#)). It receives action commands from the Agent and reports its state to Sequence Library and back to the Agent. In practice the Environment may be a continuous-time system, as long as it includes a means to execute discrete-time commands,  $a$ , and to report discrete-time sensor information,  $s$ .

The formulation of the dynamic RL problem places no constraints on the Environment. It may contain its own internal control system, stochastic elements, and learning capabilities. The Environment may do a large amount of pre-processing on its sensor data and return highly-interpreted information. Alternatively, it may return nearly raw sensor data, binned and discretized in time. It may be physical or simulated, and there are no explicit limits to the complexity it can have.

**Agent.** The Agent contains the reward function,  $g$ , and uses it to evaluate the plan candidates it receives from the Sequence Library. It executes the plans it selects by passing the corresponding actions to the Environment. The procedure the Agent follows during its operation is outlined below:

1. Define a target,  $\tau$ , consisting of the most recent  $\sigma$ .
2. Query the Sequence Library for sequences that begin with  $\tau$ ,  $\phi(\tau)$ . The set of these form  $\kappa(\tau)$ , a collection of candidate plans.
3. Select a plan to execute from  $\kappa(\tau)$ :
  - (a) Select the candidate plans that maximize the expected reward,  $\bar{r}$ , from the states that follow  $\tau$  in each  $\phi(\tau)$ .
  - (b) If there are more than one of these, select the shortest among them, that is, minimize  $n_\sigma$ .
  - (c) If there is still more than one candidate, randomly select from among the remaining candidate plans such that a single plan,  $\hat{\phi}$ , is selected.
4. Execute the actions,  $a$ , associated with each element of  $\hat{\phi}$ .
5. Return to step 1.

The Agent also passes copies of the actions it executes,  $a$ , to the Sequence Library, so that it can assemble each  $a$ - $s$  pair into a  $\sigma$ .

**Sequence Library.** The Sequence Library is at the heart of S-learning. It allows S-learning to learn from its experience, use new learning as it is gained, generalize that learning to unfamiliar situations, make predictions, and attain goals. It has two primary functions: to pass candidate plans to the Agent and to record state space trajectories as they are observed. Candidate plans,  $\phi(\tau)$  are selected on the basis of whether they begin with the target subsequence,  $\tau$ , passed in by the Agent. The set of  $\phi(\tau)$ ,  $\kappa(\tau)$ , is returned to the Agent. The process for recording newly observed states in the library is described below.

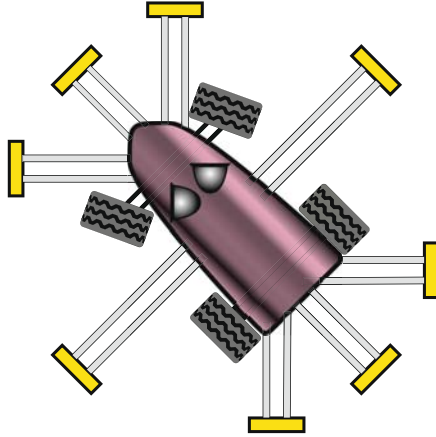
Due to the fact that S-learning is an experience-based learning algorithm, there is no distinction between memory and learning. Both are accomplished by the storage of sequences. As the Agent passes in actions,  $a$ , and the Environment passes in output states,  $s$ , the Sequence Library assembles them into  $a$ - $s$  pairs,  $\sigma$ . A working memory of the most recently observed states is maintained. Sequences,  $\phi$ , of length  $n_\sigma$  are stored in the library,  $\kappa$ . For  $\phi_j$  that begins with  $\sigma_i$ ,  $\phi_{j+1}$  will begin with  $\sigma_{i+1}$ , that is, the subsequent sequences overlap by  $n_\sigma - 1$  states. Through this accrual process,  $\kappa$  becomes the repository of the system's experience.

## 2.2 Robot Simulation

The S-learning algorithm was coded in Java and demonstrated with a simple simulated system. The simulation consisted of a mobile wheeled robot with two



light sensors and eight contact sensors. (Figure 2) The robot occupied a  $25 \times 25$  cell grid world. The robot could be positioned in the center of any one of the 625 cells in any one of the eight directions of the compass rose (up, down, right, left, and the directions offset 45 degrees from them). Contact sensors were located on the front, rear, sides, and corners of the robot. These registered whether the robot was in the border rows of the grid and in which direction were the contacted wall(s). The steering angle of the robot was also fed back with the sensory data.



**Fig. 2.** Representation of the simulated robot. The steering angle of the front wheels, wall contact in any of 8 directions, and the sensed intensity at the two light sensors were all included in the state vector information.

A light source occupied another one of the grid cells and provided input to the light sensors. The two light sensors were each oriented 45 degrees from the robot's heading, one on the right and one on the left. The intensity of the light reaching the robot was the inverse of the square of the Euclidean distance from the robot to the light source and was determined by the following equation:

$$I = \frac{1}{(x_s - x_r)^2 + (y_s - y_r)^2 + \epsilon} \quad (4)$$

where  $x_r$ ,  $y_r$ ,  $x_s$ , and  $y_s$  are the  $x$ - and  $y$ -coordinates of the robot and light source, respectively.  $\epsilon = 10^{-7}$  was added to denominator to maintain numerical stability. The off-angle sensitivity,  $\Omega$ , for each sensor was determined by the square of the cosine of the off-angle. This yielded a sensitivity of one in the direction of the sensor and a sensitivity of zero at an off-angle of  $\pm 90$  degrees. Sensitivity at greater angles was also zero. The sensed intensity for each sensor,  $\hat{I}$  was the product of the intensity and the off-angle sensitivity:

$$\hat{I} = I\Omega \quad (5)$$

The steering angle could have one of three values—straight ahead or 45 degrees to the right or left—and the robot could either drive forward or reverse. Steering and locomotion commands were incremental. A ‘steer right’ command increased the steering angle by 45 degrees unless it was already at its maximum value. ‘Steer left’ worked similarly. Movements forward and backward were made in one-cell increments. When the robot’s heading was diagonal to the grid array, movements were made to the nearest diagonal cell. Motion was executed by first rotating the robot in place by the steering angle before moving either forward or backward. When it attempted to drive into a wall head-on or into a corner, the robot was not permitted to do so and it remained where it was. When it attempted to drive into a wall at a 45 degree angle, it instead moved to the next cell along the wall and maintained its heading. The robot was also not permitted to drive backward through the light source. Simultaneous ‘steer right’ and ‘steer left’ commands resulted in no change to the steering angle, and simultaneous ‘forward’ and ‘reverse’ commands resulted in no locomotion.

**Action-State Pair Vector,  $\sigma$ .** The  $\sigma$  vector at each timestep was composed of binary elements representing the command issued and the sensory state after executing the command. The composition of  $\sigma$  is detailed in Table 1. The light sensor data was a binned  $\hat{I}$ , with the bin number given by the following:

$$b = 50 - \text{ceil} \left( \frac{1}{(\hat{I} + \epsilon)^{0.5}} \right) \quad (6)$$

The actual bin number,  $\hat{b}$ , was limited to the bins available:

$$\hat{b} = \begin{cases} 1 & \text{if } b < 1 \\ 50 & \text{if } b > 50 \\ b & \text{otherwise} \end{cases} \quad (7)$$

$\hat{b} = 1$  corresponded to very little to no light reaching the sensor, and  $\hat{b} = 50$  corresponded to very intense light exposure. Both light sensors simultaneously

**Table 1.** Composition of  $\sigma$  for the simulated robot

Sensory modality or command type	Number of elements
Command: steering	2
movement	2
Sensors: contact	8
steering angle	3
light (right)	50
light (left)	50
<b>Total:</b>	<b>115</b>

achieved  $\hat{b} = 50$  only when the robot drove forward into the light source. Vector elements corresponding to active sensor or command elements were equal to one. All others were zero.

**Reward.** The goal of the system emerged from the nature of the reward. A reward vector,  $\rho$ , was created with the same length as  $\sigma$ , such that the total reward,  $r$ , was given by the following:

$$r = \prod_i \rho_i \text{ for all } i \text{ where } \sigma_i = 1 \quad (8)$$

In this formulation,  $\rho$  served as a set of multiplicative weights for  $\sigma$ . Sensory states were rewarded or penalized by assigning higher or lower values of  $\rho$ . The  $\rho$  used in the simulation was constructed in the following way:

- $\rho$  values for steering angle positions were all set to 1, so as to neither reward nor penalize them.
- $\rho$  values for contact sensors were set to 0.7 to penalize contact with the borders of the grid.
- For the light sensors, the 50 reward vector elements that corresponded to the gradations in intensity were set according to the relation  $\rho_i = i/50$ . The most intensely sensed light produced a  $\rho$  of 1, resulting in no penalty, while the weakest sensed light produced a  $\rho$  of 0.02, a strong penalty.

**Sequence Library Creation.** At each timestep, the action that was executed and the state that resulted from that action were combined into a state-action pair,  $\sigma$ . The sequence of  $n_\sigma^{\max}$  most recently observed sequences was maintained, where  $n_\sigma^{\max}$  was the maximum sequence length, a parameter manually set in software. As described above, the longest sequence not in the library already (up to the maximum sequence length) was added to the Sequence Library. Due to the simplicity of the system, all the information necessary to make reasonably accurate predictions about the system was available at each timestep. In this case a maximum sequence length of  $n_\sigma^{\max} = 2$  was sufficient. More complex systems would benefit from a greater  $n_\sigma^{\max}$ , as it would be able to compensate somewhat for partial or noisy state information.

As sequences were added to the library, they were assigned an initial strength value,  $10^6$ . At each timestep, the strength was decreased by 1. The strength of each sequence was multiplied by 10 after each repeat observation. If strength ever decayed to 0, the sequence was dropped from the library. This provided a mechanism for rarely observed sequences to be forgotten. The deterministic nature of the simulated system did not need to make use of this (hence the large initial strength), but it is a feature of S-learning that suits it for use with more complex systems as well. It can also be seen that after several repeated observations a sequence's existence in the library would be assured for the life of the system. This is analogous to recording an experience in long-term memory.

**Action Selection.** The Agent referred to the Sequence Library to help determine which action command to send at each timestep. All sequences that began with the most recent state were used as a set of predictions. (The most recent state might be contained in multiple  $\sigma$ 's, since several actions may have resulted in that state in the system's history. Sequences beginning with all  $\sigma$ 's matching the most recent state were returned.) Each sequence represented a possible future. The Agent compared the reward at the final state of each sequence to the reward at the initial state, and the sequences with the greatest increase in reward were selected as the most promising.

The actions pertaining to each sequence defined a plan. By executing the actions in the same order, it was possible to create the same sequence of states. However it was not guaranteed to do so. Some state information, such as distance to the grid borders when not in contact with them, was not directly sensed and so introduced some variability into the effects produced by a given series of actions. Although it was a relatively minor effect with the simulated robot, with more complex systems containing more limited state information, the variability of the effects of a given action would increase greatly. The most promising sequences found in the Sequence Library represented the best case scenarios for each plan. In order to make a more informed decision, the expected value of the final reward for each plan (up to 50 of them) was calculated in the following way.

The library was queried for all the sequences starting from the most recent state and executing each plan. The final rewards for the sequences executing a given plan were averaged, weighted by the log of the strength of each sequence:

$$\bar{r} = \frac{\sum_i r_i \log(\omega_i + 1)}{\sum_i \log(\omega_i + 1)} \quad (9)$$

where  $\bar{r}$  is the weighted average reward and  $r_i$  is the reward and  $\omega_i$  is the strength associated with each sequence. One was added to  $\omega_i$  to ensure that the log remained non-negative.

$\bar{r}$  represented the expected value of the reward a given plan would produce. The plan with the highest value of  $\bar{r}$  was selected for execution, given that  $\bar{r}$  was greater than the reward at the most recent state. In the case of the simulated robot, with a maximum sequence length of two, the plan always consisted of a single action; that action command was passed to the robot at that timestep.

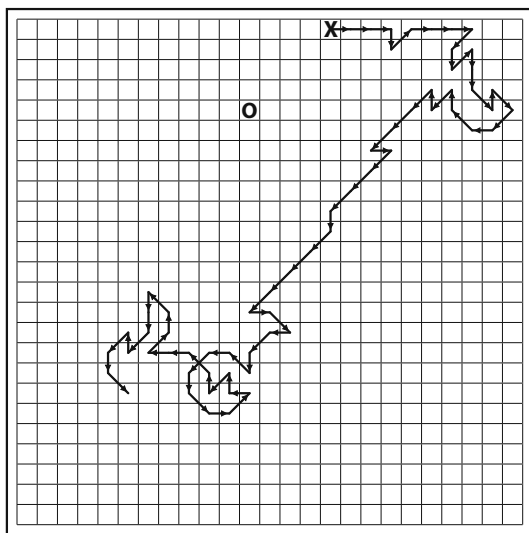
If no plans were expected to increase the reward, then the robot generated a random exploratory plan. Exploratory plans were also initiated at random intervals (on average one out of sixty timesteps). In addition, a 'boredom' condition was met if the same state was observed more than five times within the last fifty timesteps. This also resulted in the creation of an exploratory plan. Exploratory plans were of random length, up to 4 actions long. Each action was randomly generated with each of the 4 elements of the action vector having an independent, 25% chance of being active. Exploration provided some random variation to S-learning's operation, allowing it to explore its environment and avoid getting caught in highly non-optimal behavior patterns, such as infinitely-repeating cycles.

**Task Structure.** The robot was able to increase its reward by orienting itself toward the light source and approaching it. When the robot drove forward into the light source, both light sensors registered their maximum intensity, and generated the maximum reward. This was defined to be the completion of the task. After the task was completed, the robot and the light source were both randomly repositioned in the grid and the task began again. The measure of the performance in each run was the number of timesteps required to complete the task.

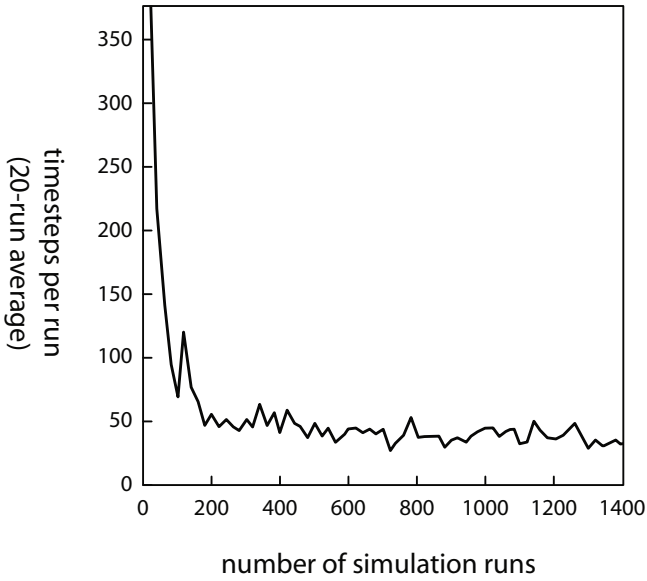
### 3 Results

On the first run the robot did not approach the light source in any direct way. The beginning of the run is shown in Figure 3. Initially, the Sequence Library was empty and all movements were random and exploratory. Learning gained during the first movements was used as soon as it was applicable. Notably, the somewhat direct descent from the upper-right region of the grid to the lower-middle region was the repeated application of single successful movement.

The earliest runs consisted mostly of exploration and were relatively lengthy. The first twenty runs averaged over 350 timesteps per run. As the Sequence Library became more complete and the state space was better explored the number of timesteps required to reach the goal decreased rapidly. (Figure 4) At 200 runs, the average number of timesteps had decreased to 50. From that point, performance continued to improve, but much more slowly. After 1400 runs, a typical run lasted 40 timesteps.

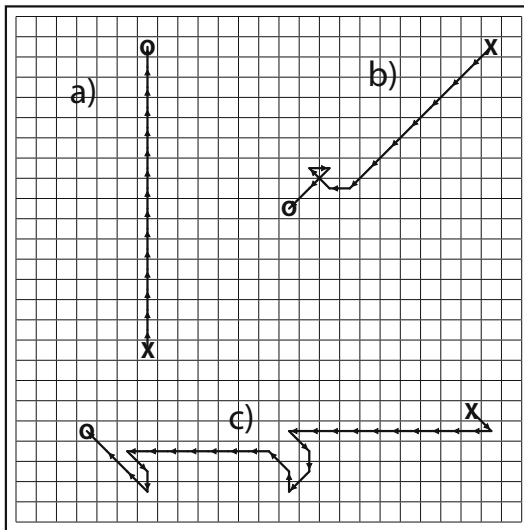


**Fig. 3.** The initial movements of a typical naïve simulation run



**Fig. 4.** Goal acquisition performance. The average performance is shown for 20-run blocks. Running on a 2.66 GHz Intel Xeon Processor with 8 GB of RAM under a 64-bit Linux, these data were generated in 16 minutes.

Later runs show a much more direct approach to the goal. Figure 5a shows an optimally direct run. Figure 5b and c show nearly direct runs that have been



**Fig. 5.** Three typical simulation runs after 1400 runs

interrupted by exploratory movements. Occasionally the robot wandered for a time before closing in on the goal, particularly when it began far from the goal, with its sensors facing away from it. This, together with exploratory interludes, caused the average performance to stay as high as it did, rather than drop to optimal levels—closer to 20.

## 4 Discussion

This work has demonstrated the implementation and operation of S-learning, a model-free learning and control approach that is fundamentally case-based. S-learning was able to learn to control a simple robot in a simple environment.

S-learning is capable of addressing some dynamic reinforcement learning problems, although it should be noted that the light-seeking robot simulated here is not one. The addition of multiple light sources of different colors, and time varying rewards associated with different colors would be an example of a dynamic RL problem. For examples of S-learning solving dynamic RL problems, see [218].

The two degree-of-freedom, non-holonomic mobile robot simulated here could be modeled with a trivial amount of effort. In fact, this model existed in the simulation in order to generate the appropriate behavior. However, S-learning didn't make use of that model (except for the structured sensory information that the simulation produced), but treated the robot system as a black box. The key aspect of S-learning's operation is that it relied only on the system's prior experience, rather than on *a priori* knowledge. This simulation does not showcase the extent of S-learning's capability. Rather, it's purpose was to detail an instantiation of the S-Learning algorithm.

### 4.1 Limitations of S-Learning

The robustness and model-independence of S-learning comes at a price. The largest cost is in long learning times. Significant training time, approximately 75,000 timesteps, was required to learn to control a relatively simple system. This raises the question of when it would be appropriate to use S-learning. In any implementation where a model is available, the trade-off between which portions to learn and control with S-learning and which to control with a more conventional model-based controller is a trade-off between learning time (short-term performance) and robustness (long-term performance). This question can only be answered based on the specific goals and constraints of each implementation.

Some of the details of S-learning's implementation are specific to the system. One of these details is the maximum sequence length,  $n_{\sigma}^{\max}$ . As described previously,  $n_{\sigma}^{\max} = 2$  was known to be appropriate to the simulation due to its determinism and simplicity. However, other systems may benefit from larger values of  $n_{\sigma}^{\max}$ . Humans' capability to remember  $7 \pm 2$  chunks of information suggest that  $n_{\sigma}^{\max} = 7$  is an estimate with reasonable biological motivation. Similarly the dynamics of sequence strength, underlying consolidation and forgetting of sequences, may need to be varied to achieve good performance on different systems. Initial tests show that the most critical design decisions in an S-learning

implementation are the discretization of sensor data and the assignment of reward vectors that produce desirable behaviors. Some primary considerations when discretizing sensors are discussed in [8,9], but additional work is required to fully identify the trade-offs involved.

## 4.2 Implications

Due to its model agnosticism, S-learning's case-based reasoning approach to robot control is potentially applicable to hard problems, such as bipedal locomotion and manipulation. In the case of locomotion, the system model can be extremely, if not intractably, complex, and environments may be completely novel. In addition, extra-laboratory environments can be harsh, and insensitivity to sensor and actuator calibration may be desirable as well. In the case of manipulation, mathematical modeling of physical contact is notoriously difficult and requires a lot of computation to perform well. It also requires high-fidelity physical modeling of the entire system, which is not possible when handling unfamiliar objects. These attributes suggest that locomotion and manipulation are two examples of hard problems to which S-learning may provide solutions.

## Acknowledgements

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

## References

1. Albus, J.: A new approach to manipulator control: Cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control* 97, 220–227 (1975)
2. Hulet, S., Rohrer, B., Warnick, S.: A study in pattern assimilation for adaptation and control. In: 8th Joint Conference on Information Systems (2005)
3. Kavvaki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4), 566–580 (1996)
4. Levinson, S.E.: *Mathematical Models for Speech Technology*, pp. 238–239. John Wiley and Sons, Chichester (2005)
5. McKinstry, J.L., Edelman, G.M., Krichmar, J.L.: A cerebellar model for predictive motor control tested in a brain-based device. *Proceedings of the National Academy of Sciences* 103(9), 3387–3392 (2006)
6. Moore, A.W., Atkeson, C.G.: The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning* 21, 199–233 (1995)
7. Morimoto, J., Zeglin, G., Atkeson, C.G.: Minimax differential dynamic programming: Application to a biped walking robot. In: *Proceedings of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pp. 1927–1932 (2003)



8. Rohrer, B.: S-learning: A biomimetic algorithm for learning, memory, and control in robots. In: Proceedings of the 3rd International IEEE EMBS Conference on Neural Engineering (2007)
9. Rohrer, B.: Robust performance of autonomous robots in unstructured environments. In: Proceedings of the American Nuclear Society 2nd International Joint Topical Meeting on Emergency Preparedness and Response and Robotics and Remote Systems (2008)
10. Sutton, R.S.: Planning by incremental dynamic programming. In: Proceedings of the Eighth International Workshop on Machine Learning, pp. 353–357. Morgan Kaufmann, San Francisco (1991)
11. Tarraf, D.C., Megretski, A., Dahleh, M.A.: A framework for robust stability of systems over finite alphabets. *IEEE Transactions on Automatic Control* (June 2008); To appear as a regular paper in the *IEEE Transactions on Automatic Control* (scheduled for June 2008)
12. Tassa, Y., Erez, T., Smart, B.: Receding horizon differential dynamic programming. In: *Advances in Neural Information Processing Systems*, pp. 1465–1472. MIT Press, Cambridge (2008)
13. Watkins, C.J.C.H.: *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England (1989)

# Quality Enhancement Based on Reinforcement Learning and Feature Weighting for a Critiquing-Based Recommender

Maria Salamó<sup>1</sup>, Sergio Escalera<sup>1,2</sup>, and Petia Radeva<sup>1,2</sup>

<sup>1</sup> Dept. Matemàtica Aplicada i Anàlisi, Facultat de Matemàtiques, Universitat de Barcelona, Gran Via de les Corts Catalanes 585, 08007, Barcelona, Spain

{maria,sergio,petia}@maia.ub.es

<sup>2</sup> Computer Vision Center, Dept. of Computer Science, Universitat Autònoma de Barcelona, 08193, Bellaterra, Spain

**Abstract.** Personalizing the product recommendation task is a major focus of research in the area of conversational recommender systems. Conversational case-based recommender systems help users to navigate through product spaces, alternatively making product suggestions and eliciting users feedback. Critiquing is a common form of feedback and incremental critiquing-based recommender system has shown its efficiency to personalize products based primarily on a quality measure. This quality measure influences the recommendation process and it is obtained by the combination of compatibility and similarity scores. In this paper, we describe new compatibility strategies whose basis is on reinforcement learning and a new feature weighting technique which is based on the user's history of critiques. Moreover, we show that our methodology can significantly improve recommendation efficiency in comparison with the state-of-the-art approaches.

## 1 Introduction

Conversational case-based recommender systems guide user through a product space, alternatively making product suggestions and eliciting user feedback [2,9,3,21]. Recommender systems can be distinguished by the type of feedback they support; examples include *value elicitation*, *ratings-based feedback* and *preference-based feedback* [22]. In this paper, we are especially interested in a form of user feedback called *critiquing* [5,14], where a user indicates a directional feature preference in relation to the current recommendation. For example, in a travel/vacation recommender, a user might indicate that she is interested in a vacation that is *longer* than the currently recommended option; in this instance, *longer* is a critique over the *duration* feature.

As part of the recommendation process, conversational systems aim to retrieve products that satisfy user preferences at each cycle. It is expected that over the course of a recommendation session, the recommender learns about user preferences, and therefore, the system could better prioritize products [15,5,13,20].

In this sense, we focus on incremental critiquing [17], which has shown to enhance the recommendation efficiency prioritizing products based on a quality measure. This quality measure is obtained by the combination of compatibility and similarity scores.

In this paper, we consider that compatibility and similarity may improve quality measure taking into account user preferences. In the literature, the compatibility score [17] is essentially computed as the percentage of critiques in the user model that a case satisfies. We argue that the moment in which a critique was made is important enough to influence the compatibility score, and thus, in the final quality measure. Note that the user increases her knowledge of the domain along cycles and her preferences are more accurate over time. In particular, previous work on this direction showed that using a simple Monte Carlo reinforcement learning strategy to compute the compatibility score obtains better case quality results [18].

Reinforcement learning (RL) [24,11] is concerned with how an agent ought to take actions in an environment. Common applications of RL techniques are related to robotics and game theory [16]. In the context of recommenders, an initial attempt to include RL techniques has been performed on web-recommendation systems where a possible analysis of finite-state Markov decision process based on pages links is possible [10]. However, in the content-based recommendation domain, it is quite difficult to infer which are possible good future actions since the environment changes with the decisions of the user. For instance, suppose that the user initially is looking for a particular video camera. The initial expectations may change with the learning process of the user while navigating in the recommendation system. While navigating, the user notices that she needs to spend more money to obtain the required product performance, and thus, critiques change. Because of this reason, instead of looking for RL techniques that predict based on how an action affects future actions, in this paper, we are going to focus on RL techniques which are based on the user specialization. This specialization is grounded on past actions, where the time of the action is closely related to the user critiquing compatibility. We review different state-of-the-art RL techniques based on Monte Carlo and Time Dynamics, and also propose two new RL techniques adapted to conversational recommender systems.

Moreover, we also argue that quality is influenced by similarity. Conversational case-based recommender systems use a similarity function (usually based on nearest neighbor rules) to recommend the most similar product at each cycle [15]. Nevertheless, similarity functions are sensitive to irrelevant, interacting, and also most preferred features [1]. This problem is well-known in Case-Based Reasoning (CBR) systems because it can degrade considerably the system performance. In order to avoid it, many similarity functions weight the relevance of features [25,12]. Previously, in the work of [19], a local user preference weighting (LW) was presented, which was shown to reduce the number of critiquing cycles. In this paper, we present a global user preference weighting (GW). This method basis on the satisfied critiques from the whole set of cases. We show how the new

weighting strategy enhances the quality, and results in a shorter session length than using the local user preference weighting.

Summarizing, this paper describes new strategies for compatibility and weighting based on user’s critiquing history for enhancing quality. The paper is organized as follows: Section 2 overviews the incremental critiquing approach as the baseline to present our methodology. Section 3 describes state-of-the-art RL approaches applied to conversational CBR and presents two new approaches adapted to critiquing. Section 4 introduces the new methodology to weight the similarity component of quality, and Section 5 presents the experimental evaluation of the presented strategies. Finally, Section 6 concludes the paper.

## 2 Background

The incremental critiquing [17] implementation assumes a conversational recommender system in the style of Entrée [4]. Each recommendation session starts with an initial user query resulting in the retrieval of a product  $p$  (also known as case) with the highest *quality*. The user will have the opportunity to accept this case, thereby ending the recommendation session, or to critique it as a means to influence the next cycle. The incremental critiquing algorithm consists of four main steps: **(1)** a new case  $p$  is *recommended* to the user based on the current query  $q$  and previous critiques; **(2)** the user *reviews* the recommendation and applies a directional feature critique,  $cq$ ; **(3)** the query,  $q$ , is *revised* for the next cycle; **(4)** the user model,  $U = \{U_1, \dots, U_i\}$ ,  $i \leq t$  is updated by adding the last critique  $cq$  and pruning all the critiques that are inconsistent with it. Finally, the recommendation process terminates either when the user retrieves a suitable case, or when she explicitly finishes the recommendation process.

This recommendation process is highly influenced by the user model  $U$  containing previous consistent critiques, which is incrementally updated at each cycle. Incremental critiquing modifies the basic critiquing algorithm. Instead of ordering the filtered cases on the basis of their similarity to the recommend case, it also computes a *compatibility* score  $C$  as follows:

$$C_t^{p'}(U) = \frac{\sum_{\forall i:(1 \leq i \leq t)} \delta(p', U_i)}{|U|} \quad (1)$$

where  $C_t^{p'}(U)$  is the *compatibility* score of candidate case  $p'$  at time  $t$  given an user model  $U$ . The satisfaction function  $\delta$  returns 1 if case  $p'$  satisfies the critique  $U_i$  or 0 otherwise, and  $|U|$  stands for the total number of critiques in the user model  $U$ . Thus, the compatibility score is essentially the percentage of critiques in the user model that case  $p'$  satisfies. Then, the compatibility score and the similarity of a candidate case  $p'$  to the current recommendation  $p$  are combined in order to obtain an overall *quality* score  $Q$ :

$$Q(p', p, U) = \beta \cdot C_t^{p'}(U) + (1 - \beta) \cdot S(p', p) \quad (2)$$

where  $S$  is the similarity function, and  $\beta$  is set to 0.75 by default. The quality score  $Q$  is used to rank the filtered cases prior to the next cycle, and the case with the highest quality is then chosen as the new recommendation.

### 3 Compatibility Using Reinforcement Learning

In this section, we analyze the compatibility component of the quality measure. As shown in [18], RL techniques can enhance compatibility efficiency. Thus, we review and propose RL techniques that are used as new compatibility scores to conversational CBR systems.

Among the different classes of RL families that exists in literature, we find Dynamic Programming Methods. These strategies are difficult to adapt to our problem since a complete and accurate model of the environment is required, and we are not able to predict future behavior of the user in the recommendation system [24]. On the other hand, Monte Carlo methods do not require a model, and are conceptually simple. Finally, temporal-difference methods (TD) also do not require a model, and are fully incremental, though they are more complex to analyze. Thus, both TD and Monte Carlo methods seem to be useful to use the user experience in order to solve the prediction problem, and retrieve the optimal product to the user reducing the number of critiquing cycles. In our case, we want to model the current compatibility  $C_t^{p'}$  of a candidate case  $p'$  at instant  $t$  based on its corresponding previous compatibility. For this task, the initial RL model for compatibility computation can be a simple Monte Carlo method [18]:

$$C_t^{p'} = C_{t-1}^{p'} + \alpha \cdot (R_t^{p'} - C_{t-1}^{p'}) \quad (3)$$

This Monte Carlo method is also called *constant- $\alpha$  MC* [24]. The term  $R_t^{p'}$  is the satisfaction of case  $p'$  at time  $t$  (i.e.,  $R_t^{p'} = 1$  if the candidate case  $p'$  satisfies the current critique, or  $R_t^{p'} = 0$  otherwise), and  $\alpha$  is a constant step-size parameter. With the simple *constant- $\alpha$  MC* of eq. (3) we can update the compatibility of a case  $p'$  at time  $t$  based on what happens to  $p'$  after current critique. Low values of  $\alpha \in [0..1]$  makes the compatibility of  $p'$  to be increased/reduced slowly, meanwhile using high values of  $\alpha$  makes the new results to affect more the compatibility of the case. We could also use incremental dynamic updates of  $\alpha$  depending of our problem domain (i.e., we could think that initial critiques of the user should have less influence that last critiques since the user still does not have a high knowledge of the recommendation environment).

In Figure 1 we show four cases and its corresponding critique satisfaction over ten cycles in an hypothetical recommender. We suppose, for this example, that each cycle  $t \in [1, \dots, 10]$  generates a new critique in our user model. The response  $R$  of each case  $p'$  at each time  $t$  is 1 if the case satisfies the current critique, or 0 otherwise. Note that all cases  $p'$  have the same number of 1's and 0's but they differ in the instant they have been satisfied. So, our expectation is that the order

of compatibility should be: first case 1, since all satisfied critiques are produced at the last cycles; next, case 4 and case 3, since both alternate 1 and 0 but the case 4 satisfies the last critique; and finally case 2 with the less compatibility since all 1's are produced at the initial cycles.

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
Case 1	0	0	0	0	0	1	1	1	1	1
Case 2	1	1	1	1	1	0	0	0	0	0
Case 3	1	0	1	0	1	0	1	0	1	0
Case 4	0	1	0	1	0	1	0	1	0	1

Fig. 1. Case base satisfaction of critiques in a toy problem

Figure 2(a) shows the RL values for *constant- $\alpha$*  MC method for the case base shown in Figure 1. Note that the final order of compatibility is the expected based on the previous criterion. We set up the compatibility at time  $t = 0$  to 0.5. The graph shows a logarithmic growing of the compatibility when satisfying critiques, and the same influence decreasing the compatibility for non satisfied cases.

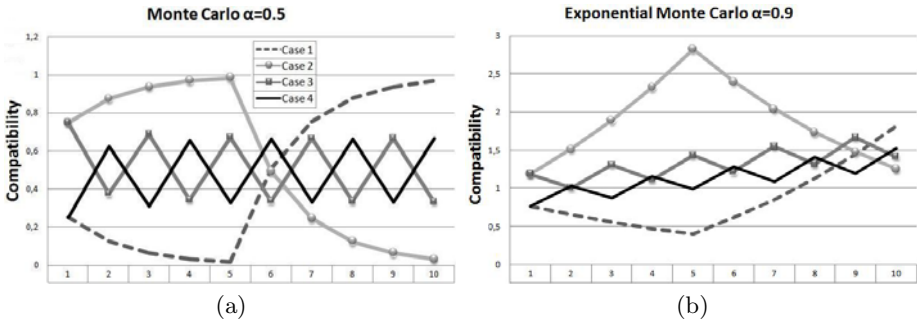


Fig. 2. (a) *constant- $\alpha$*  MC and (b) EMC numerical representation for the case base shown in Figure 1

On the other hand, we could require that changes between different results  $R_t$  modify the compatibility of  $p'$  in a different magnitude. For example, we could think that a wrong result for the case  $p'$  defined as  $R_t^{p'} = 0$  to have less influence than a good result  $R_t^{p'} = 1$ . Then, we propose the Exponential Monte Carlo (EMC) as follows:

$$C_t^{p'} = \begin{cases} C_{t-1}^{p'} + \alpha \cdot (R_t^{p'} + C_{t-1}^{p'}) & \text{if } R_t^{p'} = 1 \\ C_{t-1}^{p'} - \alpha \cdot C_{t-1}^{p'} & \text{if } R_t^{p'} = 0 \end{cases} \quad (4)$$

Note that the Monte Carlo variant EMC defined in eq. (4) varies the logarithmic increasing of the compatibility in eq. (3) by an exponential tendency for an input sequence of satisfied critiques  $R_{[1,\dots,t]}^{p'} = 1$ , as shown in Figure 2(b). With the exponential tendency the compatibility score is more significant when more critiques are satisfied in the last cycles of the recommendation process.

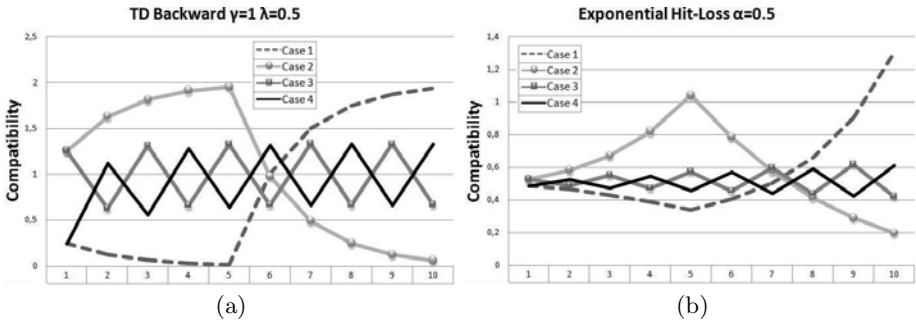
Concerning to TD methods, the Backward TD( $\lambda$ ) used in literature [24] considers an internal variable  $e_t^{p'}$  defined as the *eligibility trace* of case  $p'$  at instant  $t$ . This variable is defined as follows:

$$e_t^s = \begin{cases} \gamma \cdot \lambda \cdot e_{t-1}^s & \text{if } s \notin s_t \\ \gamma \cdot \lambda \cdot e_{t-1}^s + 1 & \text{if } s \in s_t \end{cases} \quad (5)$$

where  $s$  is the state being analyzed,  $s_t$  is the set of valid states at time  $t$ ,  $\gamma$  is the discount rate, and  $\lambda$  is related to the *eligibility trace* influence. In our case, considering the state  $s$  as a candidate case  $p'$ , we can express eq. (5) as follows:

$$C_t^{p'} = \gamma \cdot \lambda \cdot C_{t-1}^{p'} + R_t^{p'} \quad (6)$$

where  $R_t^{p'} = 1$  if the case  $p'$  satisfies the current critique and use the eligibility trace as a measure of the compatibility of  $p'$ . This method has a similar tendency than the *constant- $\alpha$*  MC, with a logarithmic increasing of the measure for an input sequence of satisfied critiques  $R_{[1,\dots,t]}^{p'} = 1$ , as shown in Figure 3(a). In this case, the desired compatibility order of the cases is also maintained. Note that the behavior of this strategy in the case of the figure is very similar to that one shown by the *constant- $\alpha$*  MC method, but working on a different compatibility range.



**Fig. 3.** (a) TD Backward and (b) EHL numerical representations for the case base shown in Figure 1

The only case from the previous methods that consider an exponential tendency for an input sequence of satisfied critiques  $R_{[1,\dots,t]}^{p'} = 1$  corresponds to the EMC RL strategy, the rest of strategies have a logarithmic compatibility

increasing. However, we can also think that in the recommendation process, as the user increases her knowledge along cycles, maybe first matches are finally not relevant meanwhile consecutive or final matches can be more confident to the user preferences. This effect could be modelled by a RL technique which changes the logarithmic increasing to an exponential one. In order to observe if this hypothesis works in conversational recommendation systems, we propose the Exponential Hit-Loss RL technique (EHL) as follows:

$$C_t^{p'} = \begin{cases} h \leftarrow h + 1, C_t^{p'} = C_{t-1}^{p'} \cdot (1 + \alpha)^{(h^{p'} + t)k} & \text{if } R_t^{p'} = 1 \\ f \leftarrow f + 1, C_t^{p'} = C_{t-1}^{p'} \cdot (1 - \alpha)^{(f^{p'} + t)k} & \text{if } R_t^{p'} = 0 \end{cases} \quad (7)$$

where  $h^{p'}$  and  $f^{p'}$  are the number of times that candidate case  $p'$  has satisfied (hit) or not (loss or fall) the critiques, respectively (for each case in the data set these values are initialized to zero at time  $t=0$ ), and  $k$  is a regularization factor (fixed to  $k = \frac{1}{2}$  in our experiments). This technique has an exponential behavior, which varies based on the amount of hit and losses in the history of each  $p'$  and the instant of time, as shown in Figure 3(b). Note that the desired compatibility order is also satisfied.

## 4 Similarity Using User Preference Weighting

As explained before, the basic idea of the recommender is to present the product that best satisfy user's preferences and we aim to do it by means of enhancing compatibility and similarity. Similarity plays, as in traditional CBR, an important role in the recommender. At each cycle, in the *standard* or the *incremental* recommendation process, the similarity between the candidate case  $p'$  to the recommended case  $p$  is computed as follows:

$$S(p', p) = \sum_{\forall f} d(p'_f, p_f) \quad (8)$$

where the similarity is the combination of distances  $d$  between the candidate  $p'$  case and the recommended case  $p$  for each feature  $f$ .

A common tendency in CBR systems is to use weighting in the similarity measure. In this sense, we propose to change the similarity measure as follows:

$$S(p', p) = \sum_{\forall f} W(p'_f) \cdot d(p'_f, p_f) \quad (9)$$

where  $W(p'_f)$  is the weight associated to the  $f$  feature of the candidate case  $p'$ .

Next, we review the local user preference weighting (LW) proposal and propose the global user preference weighting (GW). Both strategies are based on the history of critiques made by the user along the session. This history is the *user model*  $U$  defined previously, which specifies the user preferences in the current session.



#### 4.1 Local User Preference Weighting

The local user preference weighting (LW) [19] discovers the relative importance of each feature in each case as a weighting value for computing the similarity, taking into account the user preferences. LW is basically motivated by the fact that most compatible cases are quite similar on their critiqued features and differences mainly belong to those features that have not been yet critiqued. So, the aim of the local weighting method is to prioritize the similarity of those features that have not yet been critiqued. The weight of the local approach is defined over each feature  $p'_f$  of candidate case  $p'$  as follows:

$$W(p'_f) = 1 - \frac{1}{2} \left( \frac{\sum_{\forall i \in U^f} \delta(p'_i, U_i^f)}{|U^f|} \right) \quad (10)$$

where  $|U^f|$  is the number of critiques in  $U$  that refer to feature  $f$ ,  $U_i^f$  is a critique over feature  $f$ . This generates a feature weight vector for each case. A feature that has not been critiqued will assume a weight value of 1.0, and a decrement will be applied when a critique is satisfied by the case. As such, the feature weight will be proportional to the number of times a critique on this feature is satisfied by the case. However, as shown in eq. (10), weights never decrease to a 0 value. For example, in a travel vacation recommender with a user model that contains two critiques [price, >, 1000] and [price, >, 1500], a case with two features {duration, price} whose price is 2000 will have as price weight a 0.5 value because it satisfies both critiques whereas the duration weight will be 1.0 because there is no critique on this feature. It is important to recap that the key idea here is to prioritize the similarity of those features that have not yet been critiqued in a given session.

#### 4.2 Global User Preference Weighting

LW computes a feature weight vector for each case depending on the degree of satisfaction of the user critiques for this case. However, considering that the compatibility function is correctly focusing into the product space, the remaining set of cases are similarly satisfying the preferences of the user, so their feature weight vectors will also be similar and a global weighting vector is feasible.

The idea is to compute a vector of weights that will be used for the whole set of candidate cases. This weighting method only enhances the set of features that may produce better recommendation to all the cases. For each case  $p'$  in the list of candidate cases, the global weighting is defined as follows:

$$W(f) = 1 - \frac{1}{2} \left( \frac{\sum_{\forall p' \subseteq P'} \delta(p', U_i^f)}{|P'|} \right) \quad (11)$$

where  $|P'|$  is the total number of cases in the list of candidate cases. The final weight for each feature  $f$  depends on the number of cases that satisfy a critique on this feature. Similarly to LW, the most satisfied critiques will have the lowest

weight for a feature, since the system looks for prioritizing features that have not been previously critiqued. As before, weights never decrease to a 0 value. The maximum decrease is 0.5 which has experimentally shown to obtain the best performance.

The rationale behind prioritizing with the highest weight values the non critiqued features is based on the idea that they are the most important to denote differences on similarity between two cases. This happens because the compatibility score correctly focuses the product space and thus, the candidate cases are similar in the critiqued features. Consequently, the effort of the similarity is to show where the differences are in the features that the recommender is not able to focus with the compatibility because there are not critiques about them.

## 5 Empirical Evaluation

In previous sections we described different reinforcement learning measures and two weighting approaches to improve the quality measure of a conversational recommender system. We argue that quality measure may benefit from improvements on compatibility and similarity. As a result, the tendency of the quality measure is to recommend cases that better satisfy user preferences. In this section, we test our proposals using a well-known recommender data set. In particular, we look for the performance of the recommender system when using reinforcement learning techniques and also the combination of both RL and weighting proposals.

### 5.1 Setup

The evaluation was performed using the standard Travel data set (available from <http://www.ai-cbr.org>) which consists of 1024 vacation cases. Each case is described in terms of nine features. The data set was chosen because it contains numerical and nominal features and a wide search space.

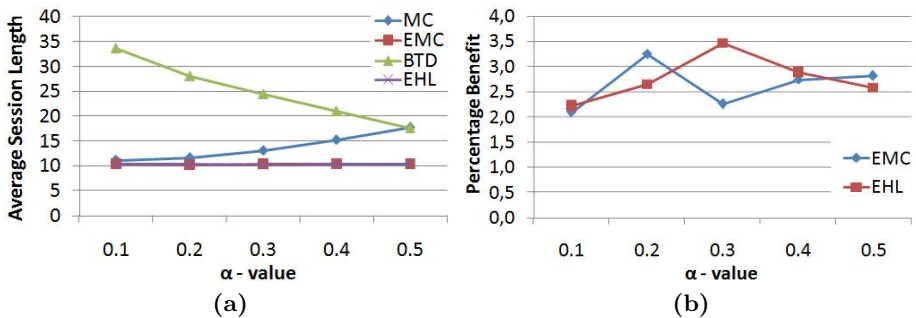
First, we evaluate the different RL measures: Monte Carlo (MC), Exponential Monte Carlo (EMC), BackwardTD (BTD), and Exponential Hit-Loss (EHL). Second, we also test the combination of RL with the weighting strategies in our recommender. The configurations analyzed are LW-MC (which corresponds to a local user preference weighting combined with a Monte Carlo compatibility), LW-BTD, LW-EHL, LW-EMC, GW-MC, GW-BTD, GW-EHL, and GW-EMC. We use incremental critiquing-based recommender (IC) [17] as baseline.

We follow the evaluation methodology similar to that one described in [23]. Accordingly, each case (which is called the 'base') in the case-base is temporarily removed and used in two ways. First, it serves as a basis for a set of queries by taking random subsets of its features. We focus on subsets of one, three, and five features to allow us to distinguish between *hard*, *moderate*, and *easy* queries, respectively. Second, we select the case that is most similar to the original base. These cases are the recommendation targets for the experiments. Thus, the base represents the ideal query for a user, the generated query is the initial

query provided by the user, and the target is the best available case for the user. Each generated query is a test problem for the recommender, and at each recommendation cycle the user picks a critique that is compatible with the known target case. We repeat each leave-one-out ten times and the recommendation sessions terminate when the target case is returned. Different statistics are also used to evaluate the statistical significance of the obtained results.

## 5.2 Reinforcement Learning Recommendation Efficiency

We analyze the recommendation efficiency —by which we mean average recommendation session length— when comparing our RL measures to incremental critiquing. RL measures need to set up a parameter  $\alpha$  that fix the learning rate (in the case of BTM we modify the parameter  $\lambda$ ). We run different variations of  $\alpha, \lambda \in [0.1, \dots, 0.5]$  in order to appreciate the influence of this parameter over each strategy. Before to compute the quality for each product, we normalize the compatibility term  $C$  so that it ranges between  $\epsilon$  and one for the lowest and highest compatibility, respectively. This is computed as  $C^{p'} = \frac{C^{p'}}{\max_{\alpha' \in P'}(C^{\alpha'})}$ , where  $C^{p'}$  is the compatibility of the case  $p'$  to be normalized. This normalization makes comparable the results obtained by the different RL strategies. Figure 4 (a) presents a graph with the evolution of the average session length for different values of  $\alpha$  and  $\lambda$ . We can see that MC and BTM present a tendency to increase and decrease the average session length when increasing  $\alpha$  and  $\lambda$ , respectively. The best configuration for MC is  $\alpha = 0.1$  and, although not shown in the graph, the best configuration for BTM is  $\lambda = 0.9$ . This large value for BTM suggests that high changes on the compatibility value may improve (reduce) session lengths. The EMC and EHL strategies consider an exponential tendency in order to make final critiques more relevant to the user preferences than the initial ones. As shown in Figure 4 (a), the exponential behavior of these strategies, in contrast to the logarithmic one of the remaining, results in shorter session lengths.

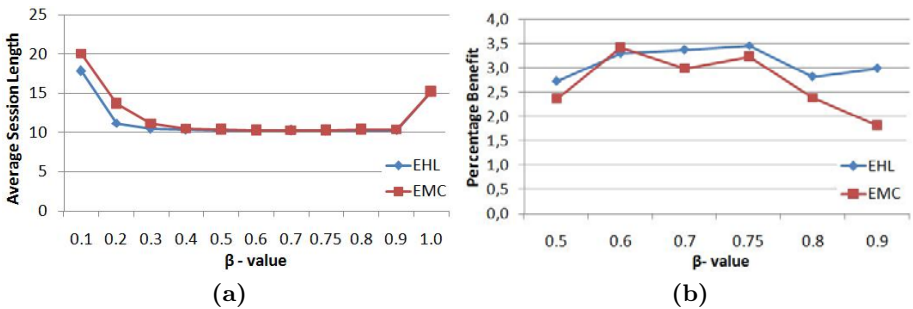


**Fig. 4.** Figure (a) corresponds to session lengths evolution for different  $\alpha, \lambda$ -values and Figure (b) represents session length benefit over incremental critiquing

We also found that the exponential EMC and EHL configurations have a more stable average session length than the rest of RL techniques for different values of  $\alpha$ , ranging from 10.37 to 10.24. In Figure 4 (b) we present EMC and EHL benefit compared to incremental critiquing. We compute the percentage benefit as  $\text{Benefit}(y, x) = \left(1 - \frac{y}{x}\right) \cdot 100$ , where  $y$  and  $x$  stands for the number of cycles of the compared strategy and IC, respectively. The EMC and EHL benefit ranges from 2% to 3.5%. Although the result seems low, we want to point out that the only difference between IC and our EMC and EHL methods is how we compute the compatibility measure. Note that in our previous work [18], we introduced the MC strategy in combination with a product recommendation strategy called *Highest Compatibility Selection* (HCS) [19], whose benefit applied together in the recommender system was around 2% to under 4%. Our new RL measures are able to obtain the same benefit by themselves without introducing the HCS methodology. Thus, we can state that EMC and EHL RL exponential strategies are able to focus on those products that best satisfy the user preferences, obtaining more accurate quality measurements.

Additionally, in Figure 5 (a) we summarize the average session lengths results over all types of queries for different variations of  $\beta$  using EMC (set up with  $\alpha = 0.2$ ) and EHL (set up with  $\alpha = 0.3$ ). Once again, the results are quite similar between EMC and EHL. Session lengths are maintained between  $\beta = 0.5$  to  $\beta = 0.9$ . It is significant that session lengths remain shorter for  $\beta = 1.0$  than  $\beta = 0.1$ . Note that  $\beta = 1.0$  means that each recommendation cycle is influenced by the compatibility measure with no similarity and a  $\beta = 0.1$  specifies that the most important role for recommendation is the similarity.

Figure 5 (b) presents EMC and EHL benefit over incremental critiquing for the  $\beta$ -values for which RL techniques obtain better results. EMC and EHL reduce the session length from nearly 2% to 3.5%. The best results are obtained for the values of  $\beta = 0.6$  and  $\beta = 0.75$ , respectively. The last value coincides with the best result obtained by the incremental critiquing. Thus, we decided to fix  $\beta = 0.75$ ,  $\alpha = 0.1$  for MC,  $\alpha = 0.2$  for EMC,  $\alpha = 0.9$  for BTD, and  $\alpha = 0.3$  for EHL in the next experiments, respectively.



**Fig. 5.** Figure (a) corresponds to session lengths evolution for different  $\beta$ -values and Figure (b) represents benefit over incremental critiquing

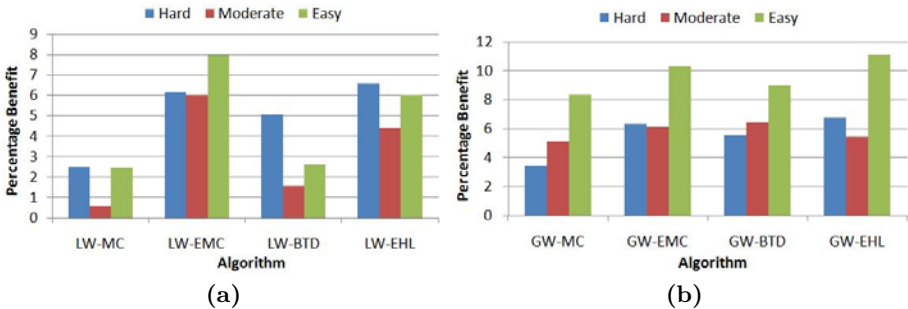
### 5.3 Quality Recommendation Efficiency

Earlier we mentioned how quality, apart from the compatibility, also uses similarity to optimize recommendations. In this section, we analyze the benefits over incremental critiquing when applying both weighting and reinforcement learning to compute the quality measure.

In Figure 6 we present the average benefit to different levels of query difficulty. Figure 6 (a) depicts the results for the combination of local weighting with RL measures. These combinations result for all algorithms tested in a reduction in session length that ranges from 0.5% up to 8%. On the other hand, see Figure 6(b), global weighting and RL measures gives the highest benefit, ranging from 3.44% to 11.13%. Combining weighting and reinforcement learning compatibility further enhances recommendation performance, resulting in better recommendation for all queries (hard, moderate, and easy).

We also statistically analyze the benefits of using our methodology instead of the standard incremental critiquing. As before, we separately evaluate local and global weighting. The algorithms analyzed are: (1) IC, LW-MC, LW-EMC, LW-BTD, and LW-HLE, and (2) IC, GW-MC, GW-EMC, GW-BTD and GW-HLE. First of all, we compute the *mean rank* ( $r$ ) of each algorithm considering all the experiments (five algorithms and three different queries for each test). The rankings are obtained estimating each particular ranking  $r_i^j$  for each query  $i$  and each algorithm  $j$ , and computing the mean ranking  $R$  for each algorithm as  $R_j = \frac{1}{N} \sum_i r_i^j$ , where  $N$  is the total number of queries. Compared with mean performance values, the mean rank can reduce the susceptibility to outliers which, for instance, allows a classifier's excellent performance on one query to compensate for its overall bad performance [6]. Second, we apply the Friedman and Nemenyi tests to analyze whether the difference between algorithms is statistically significant [7,8].

The **Friedman test**, recommended by Demšar [6], is effective for comparing multiple algorithms across multiple data sets, in our case, across multiple



**Fig. 6.** Average benefit over incremental critiquing. Each figure represents a different benefit weighting where (a) corresponds to local weighting and (b) corresponds to global weighting.

queries. It compares mean ranks of algorithms to decide whether to reject the null hypothesis, which states that all the methods are equivalent and so their ranks should be equal. The Friedman statistic value is computed as  $X_F^2 = \frac{12N}{k(k+1)} \left[ \sum_j R_j^2 - \frac{k(k+1)^2}{4} \right]$ . Since this value is undesirable conservative, Iman and Davenport proposed a corrected statistic,  $F_F = \frac{(N-1)X_F^2}{N(k-1)-X_F^2}$ .

When we apply the Friedman test in our experimental setup with five algorithms and three different queries,  $F_F$  is distributed according to the  $F$  distribution with  $(5 - 1) = 4$  and  $(5 - 1) \cdot (3 - 1) = 8$  degrees of freedom. The critical value of  $F(4, 8) = 3.83$  at the 0.05 critical level. We obtained the values of  $X_F = 11.42$  and  $F_F = 40.06$  for the local weighting rankings and  $X_F = 9.86$  and  $F_F = 9.22$  for the global weighting rankings. As the values of  $F_F$  are always higher than 3.83 we can reject the null hypothesis in both cases.

Once we have checked for the non-randomness of the results, we can perform a post hoc test to check if one of the techniques can be singled out. For this purpose we use the Nemenyi test —two techniques are significantly different if the corresponding average ranks differ by at least the critical difference value,  $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$ , where  $q_\alpha$  is based on the Studentized range statistic divided by  $\sqrt{2}$ . In our case, when comparing five algorithms with a critical value  $\alpha = 0.1$ ,  $q_{0.1} = 2.45$  for a two-tailed Nemenyi test. Substituting, we obtain a critical difference value  $CD = 3.17$ . Thus, for any two pairs of algorithms whose rank difference is higher than 3.17, we can infer —with a confidence of 90%— that there exists a significant difference between them.

The results of the Nemenyi test are illustrated in Figure 7. In the figure, bullets represent the mean ranks of each algorithm. Vertical lines across bullets indicate the 'critical difference'. The performance of two algorithms is significantly different if their corresponding mean ranks differ by at least the critical difference. For instance, Figure 7 (a) reveals that LW-EMC and LW-EHL are significantly better than IC. We cannot say the same with regard to LW-MC and LW-BTD, though. The same behavior occurs in the case of the global weighting analysis of Figure 7 (b). However, note that the global weighting of Figure 6(b)

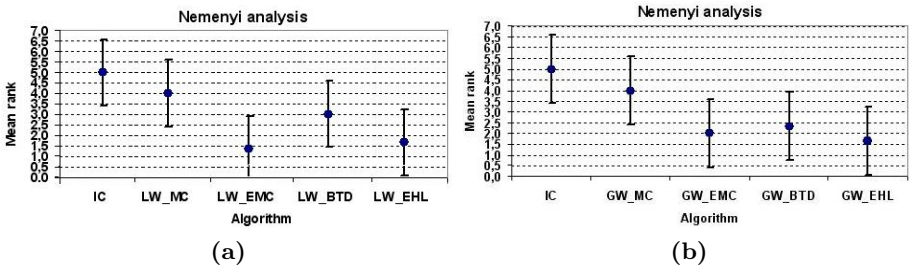


Fig. 7. Application of the Nemenyi test critical difference to algorithms mean rank considering their session length for (a) local weighting and (b) global weighting

shows more stable results for all combination of strategies than local weighting, obtaining higher benefits in terms of session length.

## 6 Conclusions

Retrieving the most suitable product for a user during a live customer interaction is one of the key pieces in conversational case-based recommender systems. Specifically, in incremental critiquing the recommendation process is primarily guided by a quality measure. In this paper we have proposed new strategies for compatibility computation and feature weighting that enhance quality. We reviewed the state-of-the-art on reinforcement learning which can be applied to conversational CBRs, and proposed two new compatibility strategies which offer better benefit in terms of session length. Concerning the similarity score, we presented a global weighting strategy, which uses a common weight over all cases based on the number of satisfied critiques. Our experiments show significantly improvements in comparison to the state-of-the-art approaches.

## Acknowledgements

This work has been supported in part by projects TIN2006-15308-C02, FIS PI061290, and CONSOLIDER-INGENIO CSD 2007-00018.

## References

1. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* 36(2), 267–287 (1992)
2. Aha, D.W., Breslow, L.A., Muñoz-Avila, H.: Conversational Case-Based Reasoning. *Applied Intelligence* 14, 9–32 (2000)
3. Burke, R.: Interactive Critiquing for Catalog Navigation in E-Commerce. *Artificial Intelligence Review* 18(3-4), 245–267 (2002)
4. Burke, R., Hammond, K., Young, B.: Knowledge-Based Navigation of Complex Information Spaces. In: *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, OR, pp. 462–468. AAAI Press/MIT Press (1996)
5. Burke, R., Hammond, K., Young, B.C.: The FindMe Approach to Assisted Browsing. *Journal of IEEE Expert* 12(4), 32–40 (1997)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal Machine Learning Research* 7, 1–30 (2006)
7. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association* 32(200), 675–701 (1937)
8. Friedman, M.: A comparison of alternative tests of significance for the problem of  $m$  rankings. *The Annals of Mathematical Statistics* 11(1), 86–92 (1940)
9. Göker, M.H., Thompson, C.A.: Personalized conversational case-based recommendation. In: Blanzieri, E., Portinale, L. (eds.) *EWCBR 2000*. LNCS, vol. 1898, pp. 99–111. Springer, Heidelberg (2000)

10. Golovin, N., Rahm, E.: Reinforcement learning architecture for web recommendations. In: Proceedings of the International Conference on Information Technology: Coding and Computing, Washington, DC, USA, vol. 2, p. 398. IEEE Computer Society Press, Los Alamitos (2004)
11. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A survey. *Journal of Artificial Intelligence Research* 4, 237–285 (1996)
12. Kohavi, R., Langley, P., Yun, Y.: The utility of feature weighting in nearest-neighbour algorithms. In: Poster at ECML 1997 (Unpublished)
13. McGinty, L., Smyth, B.: Comparison-Based Recommendation. In: Craw, S. (ed.) ECCBR 2002. LNCS, vol. 2416, pp. 575–589. Springer, Heidelberg (2002)
14. McGinty, L., Smyth, B.: Tweaking Critiquing. In: Proceedings of the Workshop on Personalization and Web Techniques at the International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco (2003)
15. McSherry, D.: Similarity and Compromise. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 291–305. Springer, Heidelberg (2003)
16. Moon, A., Kang, T., Kim, H., Kim, H.: A service recommendation using reinforcement learning for network-based robots in ubiquitous computing environments. In: IEEE International Conference on Robot & Human Interactive Communication (2007)
17. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Incremental Critiquing. In: Bramer, M., Coenen, F., Allen, T. (eds.) Research and Development in Intelligent Systems XXI. Proceedings of AI 2004, Cambridge, UK, pp. 101–114. Springer, Heidelberg (2004)
18. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Improving incremental critiquing. In: 16th Artificial Intelligence and Cognitive Science, pp. 379–388 (2005)
19. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Knowledge discovery from user preferences in conversational recommendation. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS, vol. 3721, pp. 228–239. Springer, Heidelberg (2005)
20. Shimazu, H.: ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops. *Artificial Intelligence Review* 18(3-4), 223–244 (2002)
21. Shimazu, H., Shibata, A., Nihei, K.: ExpertGuide: A Conversational Case-Based Reasoning Tool for Developing Mentors in Knowledge Spaces. *Applied Intelligence* 14(1), 33–48 (2002)
22. Smyth, B., McGinty, L.: An Analysis of Feedback Strategies in Conversational Recommender Systems. In: Cunningham, P. (ed.) Proceedings of the 14th National Conference on Artificial Intelligence and Cognitive Science, Dublin, Ireland (2003)
23. Smyth, B., McGinty, L.: The Power of Suggestion. In: Proceedings of the International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco (2003)
24. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An introduction. MIT Press, Cambridge (1998)
25. Wettschereck, D., Aha, D.W.: Weighting features. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 347–358. Springer, Heidelberg (1995)



# Abstraction in Knowledge-Rich Models for Case-Based Planning\*

Antonio A. Sánchez-Ruiz, Pedro A. González-Calero, and Belén Díaz-Agudo

Dep. Ingeniería del Software e Inteligencia Artificial  
Universidad Complutense de Madrid, Spain  
antonio.sanchez@fdi.ucm.es, {pedro, belend}@sip.ucm.es

**Abstract.** Abstraction in case-based planning is a mechanism for plan retrieval and adaptation. An abstract case is a generalization of a concrete case that can be reused in different situations to that where the original case was obtained. Additional knowledge is also required to instantiate an abstract case for a new concrete solution.

In this paper, we show how the cases built by a generative planner, that uses Description Logics to represent knowledge-rich models of the state of the world, can be automatically abstracted by using the same knowledge model. An algorithm for case abstraction is presented, along with the conditions that a new problem must fulfill for being solvable by an abstract case.

## 1 Introduction

Abstraction has played an important role in different approaches to Case-Based Reasoning, sometimes under different names such as generalized cases, prototypes or scripts. After reviewing previous work, in [1] a definition of abstract case was proposed that has been widely accepted in the CBR community: an abstract case is a particular type of generalized case, in the sense that it serves as a substitute for a set of concrete cases, that, in addition, is represented through a somehow simplified (abstracted) formalism with regard to that used in concrete cases.

Abstraction plays also a key role within the field of Knowledge Representation [2]. The *is-a* relationship is at the core of frame systems and semantic nets, although not very well formalized. Actually, modern knowledge representation technologies such as Description Logics (DLs) have been successful in replacing previous approaches due to its well-founded semantics for abstraction in concept hierarchies. Concept abstraction can be reduced to logic entailment, which can be computed in restricted subsets of first-order predicate logic such as DLs.

In spite of their common roots (Schanck's scripts were a sophisticated form of semantic nets), work on CBR has been only lightly connected to advances in Knowledge Representation technology. Abstraction in CBR systems has mainly

---

\* Supported by the Spanish Ministry of Science and Education (TIN2006-15140-C03-02, TIN2006-15202-C03-03).

**Table 1.** Concept constructors

Name	Syntax	Name	Syntax
conjunction	$C \sqcap D$	disjunction	$C \sqcup D$
negation	$\neg C$	nominal	$\{a\}$
existencian restr.	$\exists r.C$	universal restr.	$\forall r.C$
at-least restr.	$(\geq nrC)$	at-most restr.	$(\leq nrC)$

used simple taxonomies, in the form of type hierarchies where a set of values (labels) can be abstracted by its type (another label). Our goal is to bring expressive knowledge representation formalisms into Case-Based Planning systems.

It has been identified as a challenging area of research the integration of existing planning algorithms with rich representations of domain-specific knowledge about planning tasks and objectives, actions and events as part of a plan, and the complex relations among them [3]. Our work focuses on the representation of the domain vocabulary and the planning state using Description Logics. This way the planner can deal with incomplete knowledge, perform complex inferences and check the consistence of the model during the whole planning process. In addition, such a planner can work on reusable knowledge models (ontologies) that have not been built specifically for planning.

In this paper, we show how plans generated by such a planner can be automatically abstracted through the knowledge included in the domain ontology used to represent the world state in the planner. At the same time, the algorithm used for abstracting a plan serves also to index the resulting abstract case. And, finally, that same declarative knowledge will guide the reuse process by guiding the specialization of an abstract case to solve a new concrete problem.

The rest of the paper runs as follows. Next Section briefly describes the basic ideas of knowledge representation in Description Logics and introduces an example domain that will be used along the paper. Section 3 presents the generative planner that is used to generate cases in our approach. Next, Sections 4 and 5 describe the abstraction process and how abstract cases are reused to solve new concrete problems. Section 6 discusses the update semantics behind operators. Finally, last Section presents related work and concludes the paper.

## 2 Abstraction in Description Logics

Description Logics (DLs) [4] are expressive subsets of First Order Logic with good reasoning properties that are specially designed to represent structured knowledge. DLs represent knowledge using *concepts*, *roles* and *individuals*. Concepts represent categories or types described by means of potential complex formulas, roles are binary relations, and individuals (concept instances) are entities that represent objects in the domain. The main property is the *is-a* relation that defines a hierarchy in which general concepts subsume more specific ones.

Concepts and roles are defined using a predefined set of *constructors*, that determine the expressive power of the DL. Table 1 shows a standard set of constructors that can be used to inductively build complex concepts from a set  $N_C$  of *concept names*, a set  $N_R$  of *role names*, and a set  $N_I$  of *individual names*.

A *Knowledge Base* has two different parts: the TBox or *terminological component*, and the ABox or *assertional component*. Intuitively, the TBox describes the domain vocabulary and a set of domain axioms that will be used to infer new knowledge and to check the model consistency. Note that using the concept constructors we can define complex vocabularies with several dependencies between terms. While the TBox represents fixed information about the domain, the ABox represents knowledge about the current state that may change.

Let  $A \in N_C$  be a concept name,  $C$  a general concept, then a *TBox* is a collection of *axioms* of the form  $A \sqsubseteq C$  ( $A$  is subsumed by  $C$ ) or  $A \equiv C$  ( $A$  is equivalent to  $C$ ).

An interpretation  $I$  gives meaning to concepts, roles and individual names by assigning them a denotation (subset of the domain world). The semantics of TBox definitions is the usual: we say that an interpretation  $I$  is a *model* of a TBox  $K_T$  ( $I \models K_T$ ) iff it *satisfies* all its axioms.

Let  $C$  be a concept,  $r$  a role, and  $a$  and  $b$  individuals, then an *ABox* is a collection of *assertions* of the form  $C(a)$ ,  $r(a, b)$  and  $\neg r(a, b)$  representing respectively that  $a$  is an instance of  $C$ ,  $a$  and  $b$  are and are not related using  $r$ . We say that an interpretation  $I$  is a *model* of an ABox  $K_A$  iff it satisfies all its assertions.

Finally, a Knowledge Base (KB from now on)  $K = (K_T, K_A)$  is a tuple where  $K_T$  is a TBox and  $K_A$  is an ABox. We say that a KB  $K$  is *consistent* if there exists an interpretation  $I$  that is a model of both  $K_T$  and  $K_A$ .

It is important to emphasize that reasoning in DLs is made under the *Open World Assumption* (OWA), i.e., the default truth value of every assertion is unknown. In other words, the ABox is only a partial description of the state and it admits several different models, one for each possible concrete state matching the partial description. We can take advantage of this feature to reason in incomplete knowledge scenarios that, due to their complexity or uncertainty, we are not able to describe completely. The more knowledge we provide to the reasoner the more inferences will be able to do, but new knowledge will never invalidate any of the previous inferences.

We can perform queries to retrieve information from a KB. Let  $V$  be a set of variable names,  $C$  a concept,  $r$  a role, and  $x, y \in V \cup N_I$  individuals or variable names, then a *conjunctive query*  $Q$  is a conjunction of atoms of form  $C(x)$ ,  $r(x, y)$  or  $\neg r(x, y)$ . The result of executing a query on a KB  $K$  is a set of substitutions  $\{\theta_1 \dots \theta_n\}$  that bind variables to individuals such that  $K_A \models \theta_i Q$ , where  $\theta_i Q$  are the assertions that result from applying the substitution to the query. We say that the query is *satisfiable* if the substitution set is not empty.

In summary, DLs are a powerful tool to represent knowledge and to reason at different levels of abstraction because (a) we can define a domain hierarchy of

---

<sup>1</sup> From the TBox we can explicitly name RBox as the finite set of role inclusion axioms.

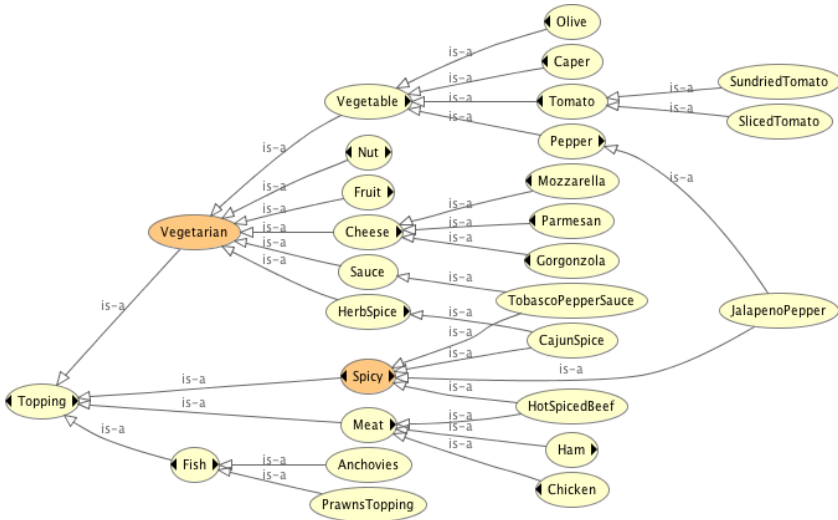


Fig. 1. Some toppings from the Pizza Restaurant ontology

concepts to describe both, general and concrete scenarios, and (b) any reasoning made in an abstract scenario will be still valid in a more concrete one.

## 2.1 The Pizza Restaurant Domain Example

In this section we introduce a domain about a pizza restaurant that will be used in the rest of the paper<sup>2</sup>. The vocabulary of this domain, formalized in OWL-DL [5], is an extension of the ontology described in the well-known OWL tutorial of Manchester University [6]. Figure 1 shows a very small part of this KB that counts with more than 130 different concepts about pizzas, toppings, bases, etc.

Toppings are classified using their types (meat, fish, cheese, fruit, etc) and their spiciness (mild, medium, hot). Pizzas, on the other hand, are classified according to their ingredient types (meaty, vegetarian, spicy ...), pizza base (thin, deep pan, etc), size (small, regular, familiar) and number of toppings. Besides, the KB contains some standard pizzas that can be found in any restaurant like *FourSeasons*, *Margherita* or *Napoletana*.

The concept hierarchy provides a large vocabulary to describe situations with different levels of detail. For example, *MushroomTopping* or *DeepPanBase* represent concrete types of ingredients and bases, while *VegetarianPizza* or *SpicyTopping* are abstract concepts that will be useful to describe situations in which we do not know or we do not care about the real ingredient types.

Next, we show some axioms from the TBox as an example of possible concept formulas. Very briefly, axiom (1) sets the different types of spiciness, (2) defines

<sup>2</sup> Available in <http://gaia.fdi.ucm.es/people/antonio/data/iccbr09/pizzeria.owl>

what a spicy topping is, (3) and (4) describe two concrete topping types, (5) defines that a pizza is spicy if it has at least one hot topping, (6) is similar for pizzas having meat, (7) says that a pizza is interesting when it has at least 3 toppings, and at last, (8) defines a *margherita* pizza as a pizza with tomato and cheese.

$$\text{Spiciness} \equiv \text{Hot} \sqcup \text{Medium} \sqcup \text{Soft} \quad (1)$$

$$\text{Spicy} \equiv \text{Topping} \sqcap \text{Hot} \quad (2)$$

$$\text{Olives} \sqsubseteq \text{Vegetable} \sqcap \text{Soft} \quad (3)$$

$$\text{Mozzarella} \sqsubseteq \text{Cheese} \sqcap \text{Soft} \quad (4)$$

$$\text{SpicyPizza} \equiv \text{Pizza} \sqcap \exists \text{hasTopping.Spicy} \quad (5)$$

$$\text{MeatyPizza} \equiv \text{Pizza} \sqcap \exists \text{hasTopping.Meat} \quad (6)$$

$$\text{InterestingPizza} \equiv \text{Pizza} \sqcap (\geq 3 \text{hasTopping}) \quad (7)$$

$$\begin{aligned} \text{Marguerita} \equiv \text{Pizza} \sqcap (= 2 \text{hasTopping}) \sqcap \exists \text{hasTopping.Cheese} \sqcap \\ \exists \text{hasTopping.Tomato} \end{aligned} \quad (8)$$

We can describe different states using ABox assertions, for example:

$$\begin{aligned} \text{SpicyPizza}(p), \text{MeatyPizza}(p), \text{hasTopping}(p, t1), \text{Olives}(t1), \\ \text{hasTopping}(t2), \text{Mozzarella}(t2) \end{aligned}$$

It is important to remember that due to OWA this is just a *partial description* of a real state and, therefore, it only says that there is a meaty and spicy pizza that has olives and mozzarella. Nothing else is assumed, i.e., the pizza could have any type of base or even other toppings different from  $t1$  and  $t2$ . Actually, from those assertions the reasoner will infer that the pizza must have *at least* another (unknown) topping of type spicy meat, because neither olives nor mozzarella are spicy or meat. This way, the pizza must have 3 or more toppings and will be classified as an instance of *InterestingPizza*. This is just an example of the complex inferences that the KB can produce.

Finally, by way of example we show a query to retrieve all the pizzas having at least one vegetarian topping:

$$\text{Pizza}(?x), \text{hasTopping}(?x, ?y), \text{Vegetarian}(?y)$$

### 3 Generative Planning Using DLs

The idea of using the expressivity power of DLs in complex planning scenarios is not new [3], but it has acquired more importance during the last years with the development of the Semantic Web and the problem of composing Web Services [7]. In this section we introduce a planning model that uses DLs in order to deal with incomplete knowledge and to perform complex inferences during the planning process. In particular, we propose to use DLs to describe: first, a rich domain

vocabulary that can be used to define planning actions and complex domain constraints; and second, incomplete planning states (or *abstract states*) that represent, in a compact way, all the states that fit in the given partial description.

A *planning domain*, in our approach, consists of a knowledge base  $K = (K_T, K_A)$  and a set of planning operators. The KB describes the domain vocabulary and a set of axioms that will be used to infer new knowledge and to detect inconsistencies during the planning process. Operators, on the other hand, describe atomic actions using the domain vocabulary. A *planning problem* consists of an initial planning state and a planning goal. A *planning state* is another KB  $S = (S_T, S_A)$  with  $S_T = K_T$  and  $S_A = K_A \cup K'_A$ , i.e., the domain KB plus a new set of assertions that define a particular instantiation in a specific instant of time. Due to the OWA, the state is an incomplete or partial description of the real state, that is unknown. The *planning goal* is represented by means of a conjunctive query that must be satisfied in the goal state. To solve the planning problem, we must find a *valid plan*, that is, a sequence of ground operators that makes the system evolve from the initial state to another state which satisfies the goals.

A *planning operator*  $O = (O_{pre}, O_{post})$  is a tuple where:

- $O_{pre}$ , the precondition, is a conjunctive query.
- $O_{post}$ , the postcondition, is another conjunctive query.
- both,  $O_{pre}$  and  $O_{post}$ , share the same set of variables.

The informal semantics of the operator is: the operator is applicable if its precondition is satisfiable in the current state, and after applying it the system will change to another state that satisfies the postcondition. Note that this approach differs from the classic STRIPS [8], in which operators have to specify the assertions that must be added to/removed from the current state. It is well known that, as the domain becomes more complex, it is an increasingly difficult task to explicitly list all the possible consequences that might occur, depending upon the details of the situation in which the action is executed (*ramification problem* [9]). Our approach is more declarative, the planner has to decide how to update the state to satisfy the postcondition.

Let  $S = (S_T, S_A)$  be the current state and  $O = (O_{pre}, O_{post})$  a planning operator:

- we say that  $O$  is *applicable* in  $S$  iff  $O_{pre}$  is satisfiable in  $S$ , i.e., exists a substitution  $\theta$  such that  $S \models \theta O_{pre}$ .
- the result of applying  $O$  in  $S$  will be another state  $S' = (S_T, S'_A)$  such that  $S' \models \theta O_{post}$  and  $S'_A$  represents a *minimal change* w.r.t.  $S_A$ .

Unfortunately, there is no consensus about the meaning of *minimal change*, and there exist different semantics regarding the effects of applying an operator [10, 11, 12]. Actually, there is no consensus on a single correct solution even for propositional KBs. Katsuno and Mendelzon [13] propose some basic characteristics an update operation should have, but the update semantics developed in literature do not agree on these properties as examined in detail by Herzig and Rifi [14].

In this paper we will assume an update operator based on the basic WIDTIO approach (When In Doubt Throw It Out [15]). According to this approach, if we reach an inconsistent state after applying an operator, we will remove assertions to make the new state consistent again. Obviously, we cannot remove assertions that are necessary to satisfy the postcondition neither assertions that are part of the domain KB (domain constraints). Note that, usually, there will be several different ways to solve the inconsistencies removing different sets of assertions, and therefore, the application of an operator may lead to different states. Intuitively, those states represent different ways of applying the operator. The concrete semantics used in our approach is described later in section 6.

Finally, a *plan* is a sequence of pairs  $(O_i\varphi_i)$  where  $O_i$  represent ground planning operators and  $\varphi_i$  sets of assertions that must be removed in order to make the new state consistent. A plan represents a generic way to solve problems in which the initial state is more concrete than the initial state used to generate the plan.

### 3.1 Planning in the Pizza Restaurant Domain

In this section we use the pizza restaurant ontology to define a basic planning domain. Besides the ontology, that sets the domain vocabulary and constraints (or axioms), we need to define planning operators. As one might expect, planning operators will be related with the creation of pizzas. In particular there is one operator to select the pizza base (*selectBase*) and some others to add toppings depending on the number of toppings in the current pizza (*addTopping0*, *addTopping1*, ...). Due to the OWA we need to assert explicitly the number of toppings in the resulting pizza to prevent the existence of other “unknown” toppings.

```
oper : selectBase
pre :NewItem(?pizza),Base(?base),Available(?base)
post : Pizza(?pizza),hasBase(?pizza,?base),Pizza0Toppings(?pizza)
```

```
oper : addTopping0
pre : Topping(?top), Available(?top), Pizza0Toppings(?pizza)
post : hasTopping(?pizza,?top), Pizza1Toppings(?pizza)
```

...

Next, we propose a very simple planning problem consisting in the creation of a margherita pizza with thin base. We can describe this goal using the following query  $\{Margherita(?x),hasBase(?x,?y),ThinBase(?y)\}$ . Let's suppose that the initial state is:

```
NewItem(p1), Available(b1), ThinBase(b1), Available(t1),
SundriedTomato(t1), Available(t2), Cheese(t2), Available(t3),
Vegetarian(t3), Spicy(t3), Mushrooms(t4)
```

Note that we do not know what type of cheese  $t2$  is, neither the real type of  $t3$  nor even if  $t4$  is available, but that information is not needed to solve the problem. Next we show a valid solution plan, where assertions between brackets are the ones that must be deleted after applying the operator in order to solve inconsistencies:

```
selectBase{?pizza = p1, ?base = b1}[NewItem(p1), Available(b1)],
addTopping0{?pizza = p1, ?top = t1}[Available(t1), Pizza0Toppings(p1)],
addTopping1{?pizza = p1, ?top = t2}[Available(t2), Pizza1Toppings(p1)]
```

## 4 Creating Abstract Cases

The planning model introduced in the previous section is able to solve abstract problems, i.e., problems where the initial state and the goal state are partially described. Plans generated will be useful to solve other problems having a more specific initial state and a more generic goal. From this point of view any pair problem-solution is a potential abstract case.

A case  $C$  is a tuple  $(C_S, C_G, C_P)$  where:

- $C_S$ , the case *precondition*, is a conjunctive query.
- $C_G$ , the case *goal*, is another conjunctive query.
- $C_P$  is a generalized *plan*, i.e., a sequence  $\{O_1\varphi_1 \dots O_n\varphi_n\}$  that may use variables of  $C_S$ .

The intended meaning of a case is that we can reuse a pre-computed plan if the current state satisfies  $C_S$  and the goal is  $C_G$  or a more general goal. Intuitively, we say that a query  $Q_1$  is more general than other query  $Q_2$  ( $Q_2 \sqsubseteq Q_1$  or  $Q_1$  *subsumes*  $Q_2$ ) if every solution of  $Q_2$  is also a solution of  $Q_1$  [7].

We will try to generalize the problem-solutions obtained from the generative planner before storing the corresponding cases in the case base, in order to create cases applicable in a broader spectrum of situations. The algorithm of generalization takes advantage of the domain concept hierarchy, using the following intuitive idea: parents of a concept describe more general categories and the children more specific ones.

We can generalize a problem  $(S_0, G)$  and its solution  $P$  using the following steps:

1. explicitly add to the initial state every assertion  $\alpha$  such that  $S_0 \models \alpha$ .
2. remove any assertion from the initial state that is not necessary to solve the problem using  $P$ .

The first step just adds to the initial state every assertion that can be inferred (using parent concepts and roles in the hierarchy), so we obtain an equivalent initial state. The second step tries to remove assertions that are not necessary, but now, as a result of the previous step, to delete an assertion does not imply to lose all the information that was deduced using it (because that information



is still asserted). Note that to perform this generalization we need to execute the plan each time we remove an assertion from the initial state in order to ensure that the plan is still valid. This process can easily be improved with a proper analysis, but to simplify we will suppose that the generation of cases is performed off-line and performance is not a issue.

#### 4.1 An Example of Case in the Pizza Restaurant Domain

We are going to create a generalized case using the problem about the margherita pizza that was solved in section 3.1. First we represent the initial state using explicitly all the assertions that can be inferred:

*NewItem(p1), Available(b1), ThinBase(b1), Base(b1), Available(t1),  
SundriedTomato(t1), Tomato(t1), Vegetable(t1), Vegetarian(t1),  
Topping(t1), Available(t2), Cheese(t2), Vegetarian(t2), Topping(t2),  
Available(t3), Vegetarian(t3), Spicy(t3), Topping(t3), Mushrooms(t4),  
Vegetable(t4), Vegetarian(t4), Topping(t4)*

Then we remove all the assertions that are not necessary to solve the problem, and the remaining initial state is:

*NewItem(p1), Available(b1), ThinBase(b1), Available(t1),  
Tomato(t1), Available(t2), Cheese(t2)*

Finally, we replace individuals with variables obtaining the following case:

$C_S : \text{NewItem}(?x1), \text{Available}(?x2), \text{ThinBase}(?x2), \text{Available}(?x3),$   
 $\text{Tomato}(?x3), \text{Available}(?x4), \text{Cheese}(?x4)$   
 $C_G : \text{Margherita}(?x), \text{hasBase}(?x, ?y), \text{ThinBase}(?y)$   
 $C_P : \text{selectBase}\{?pizza = ?x1, ?base = ?x2\}[\text{NewInd}(?x1), \text{Available}(?x2)],$   
 $\text{addTopping0}\{?pizza = ?x1, ?top = ?x3\}[\text{Available}(?x3), \text{Pizza0Toppings}(?x1)],$   
 $\text{addTopping1}\{?pizza = ?x1, ?top = ?x4\}[\text{Available}(?x4), \text{Pizza1Toppings}(?x1)]$

Intuitively, the knowledge represented in this case is: to build a margherita pizza with thin base you only need a thin base and two toppings of types tomato and cheese; take the base, add the tomato and finally add the cheese.

#### 4.2 Case Base Indexing and Retrieval

We need to structure the case base in order to retrieve *relevant* cases quickly. In our context, by relevant we mean cases in which the precondition is satisfied in the current initial state and the goal describes a more specific state than the goal of the current problem.

The Case-Based Planner uses two different KBs to represent the planning state and to store cases. The KB of cases contains a new concept *Case* and a new role *hasSK* to relate each case to its information. In particular, we will store the cases indexed by their goals, replacing the variables in the goal query with new skolem individuals, and using the role *hasSK* to relate each case with all its skolem individuals.

For example, we can store the *margherita case* of the previous section using the following assertions. The first line results from replacing variables in the goal query with new individuals *sk1* and *sk2*, and the second line sets that this information belongs to the case *c1*.

$$\begin{aligned} & Margherita(sk1), hasBase(sk1, sk2), ThinBase(sk2) \\ & Case(c1), hasSK(c1, sk1), hasSK(c1, sk2) \end{aligned}$$

Let's suppose that now we have to solve a new problem consisting in building a vegetarian pizza with thin base. We can retrieve the cases whose goal is more specific using the following query. Note that *c1* will be bound to variable *?z* because *Margherita* pizza is inferred to be a *VegetarianPizza* since all its toppings are vegetarian.

$$\begin{aligned} & VegetarianPizza(?x), hasBase(?x, ?y), ThinBase(?y) \\ & Case(?z), hasSK(?z, ?x), hasSK(?z, ?y) \end{aligned}$$

Once all those cases have been retrieved we need to check which ones of them are applicable in the current planning state. This is easily accomplished executing their preconditions in the state KB and selecting those cases that have at least one solution.

## 5 Case Reuse

We say that a case  $C = (C_S, C_G, C_P)$  can be used to solve a problem  $P = (S_0, G)$  iff:

- $C_G \models G$ , i.e., the goal of the problem subsumes the goal of the case.
- $S_0 \models C_P$ , i.e., the case precondition is satisfiable in the initial state.

The solution stored in the case is a valid solution for the problem, although it is possible that during the plan execution we will have to remove some extra assertions to reach consistent states. The execution of the case precondition in the current initial state provides the required bindings to execute the plan in the new context.

For example, let's suppose that we have to solve a problem in which the goal is to make a vegetarian pizza using the following ingredients:

$$\begin{aligned} & NewItem(np1), ThinBase(nb1), Available(nb1), SlicedTomato(nt1), \\ & Available(nt1), Parmesan(nt2), Available(nt2), Beef(nt3), Available(nt3) \end{aligned}$$

Using the goal of the problem as a query in the KB of cases we retrieve the case *c1* because:

$$\text{Margherita}(\?x), \text{hasBase}(\?x, \?y), \text{ThinBase}(\?y) \sqsubseteq \text{VegetarianPizza}(\?x)$$

Besides, the current initial state satisfies the precondition of the case using the substitution  $\{\?x1 = np1, \?x2 = nb1, \?x3 = nt1, \?x4 = nt2\}$ . Finally, a solution of the problem is computed from the case solution:

$$\begin{aligned} & \text{selectBase}\{\?pizza = np1, \?base = nb1\}[\text{NewInd}(np1), \text{Available}(nb1)], \\ & \text{addTopping0}\{\?pizza = np1, \?top = nt1\}[\text{Available}(nt1), \text{Pizza0Toppings}(np1)], \\ & \text{addTopping1}\{\?pizza = np1, \?top = nt2\}[\text{Available}(nt2), \text{Pizza1Toppings}(np2)] \end{aligned}$$

Note that during the case reuse we have implicitly replaced *SundriedTomato* with *SlicedTomato* and *Cheese* with *Parmesan*. This is a very simple example but in complex scenarios we can perform more interesting adaptations. The important part is that all the process of generalization and concretion are based on the inferences that the reasoner is able to do from a declarative description of the domain.

## 6 Operator Semantics

We have not described the real semantics behind an operator execution until now, leaving the intuitive notion of deleting problematic assertions when we reach an inconsistent state during the planning process. In this section we provide the details required in order to understand how operators are applied and why a generalized plan can be used to solve more specific problems.

First of all, in order to minimize the destructive effects of the WIDTIO approach, when we remove an assertion from the current state we do not want to lose the information that could be inferred from it. We can obtain this effect either using states that explicitly contain all the possible inferred assertions or explicitly adding the adequate information when we remove an assertion. We consider from now on that states explicitly contain all the assertion that can be deduced.

Let  $S_T$  be a satisfiable TBox, then we say that a set of assertions  $\phi$  is an *inconsistency* w.r.t  $S_T$  iff the KB  $(S_T, \phi)$  is inconsistent and for any  $\alpha \in \phi$   $(S_T, \phi \setminus \{\alpha\})$  is consistent. An inconsistency represents a minimal set of assertions that cause a KB to be inconsistent, and to solve it we only have to remove one of them. Note that, in general, there will be several inconsistencies in an inconsistent KB. Besides, if some inconsistencies share an assertion  $\alpha$ , we can solve them just removing  $\alpha$  from the KB. We use  $\text{inc}((S_T, S_A))$  to represent the set of inconsistencies in  $(S_T, S_A)$ .

Let  $S = (S_T, S_A)$  be an inconsistent state, we say that a set of assertions  $\varphi$  is a *inconsistency solver* of  $S$  iff  $(S_T, S_A \setminus \varphi)$  is consistent. We say that  $\varphi$  is a *minimal inconsistency solver (mis)* of  $S$  iff  $\varphi$  is an inconsistency solver and there is no other inconsistency solver  $\psi \subset \varphi$ .

Now we can define how operators update the planning state. Let  $S = (S_T, S_A)$  be a consistent state and  $O = (O_{pre}, O_{post})$  a planning operator, then  $apply(S, O)$  is the set of possible next states that is computed as follows:

- if  $S_A \cup O_{post}$  is consistent w.r.t  $S_T$  then  $S'_A = S_A \cup O_{post}$  and  $apply(S, O) = \{(S_T, S'_A)\}$ .
- in other case,  $S'_A = S_A \cup O_{post} \setminus \varphi^i$  where  $\varphi^i$  is a *mis* of  $(S_A \cup O_{post})$ , and  $apply(S, O) = \{(S_T, \bigcup S'_A)\}$ .

Using *mis* we prevent the deletion of too many assertions, for example, deleting all the ABox assertion but  $O_{post}$ . The union of an operator and a *mis* univocally determines the next state:  $apply(S, O, \varphi) = (S_T, S_A \cup O_{post} \setminus \varphi)$ .

A *plan*  $P = (O_1\varphi_1 \dots \varphi_n\varphi_n)$  is a sequence of operators and *mis*. We say that the plan  $P$  is *executable* from the state  $S_0 = (S_T, S_A^0)$  if:

- $S_0 \models O_{pre}^1$
- for all  $S_i = apply(S_{i-1}, O_i, \varphi_i)$  holds that  $S_i \models O_{pre}^{i+1}$

The intuitive idea behind plan reuse is that if an operator  $O$  transforms the state  $S$  into  $S'$ , and another state  $R$  is more specific than  $S$ , then there must be at least one way of applying  $O$  in  $R$  that leads to another state  $R'$  more specific than  $S'$ . Next we formalize it and show the main ideas behind the demonstration.

**Proposition 1.** *Let  $S = (K_T, S_A)$  be a state,  $O = (O_{pre}, O_{post})$  an operator applicable in  $S$ ,  $\varphi$  a *mis* of  $(S_A \cup O_{post})$ ,  $R = (K_T, R_A)$  another state more specific than  $S$  ( $R_A \models S_A$ ), then:*

1. *operator  $O$  is applicable in state  $R$ .*
2. *there exists a set of assertions  $\psi$  *mis* of  $(R_A \cup O_{post})$ , such that  $\varphi \subseteq \psi$  and  $apply(R, O, \psi) \models apply(S, O, \varphi)$ .*

First part is trivial because  $R_A \models S_A$  and  $S_A \models O_{pre}$ .

To demonstrate the second part we will build the set  $\psi$ . It is easy to see that  $(R_A \cup O_{post}) \models (S_A \cup O_{post})$  because we add the same information to both ABoxes, and then  $inc(S_A \cup O_{post}) \subseteq inc(R_A \cup O_{post})$ . Let  $newInc = inc(R_A \cup O_{post}) \setminus inc(S_A \cup O_{post})$  be the set of new inconsistencies, then every inconsistency  $\phi \in newInc$  has at least one assertion  $\alpha \in (R_A \cup O_{post}) \setminus (S_A \cup O_{post})$ , i. e., an assertion that depends on  $(R_A \setminus S_A)$ . Then there exist a set of assertions  $\mu$  *mis* of  $(R_A \cup O_{post}) \setminus \varphi$  that is empty or only contains assertions of  $(R_A \cup O_{post}) \setminus (S_A \cup O_{post})$ . We build  $\psi = \varphi \cup \mu$ .

**Proposition 2.** *Let  $S_0$  be a state and  $(O_1\varphi_1 \dots O_n\varphi_n)$  an applicable plan such that  $S_i = apply(S_{i-1}, O_i, \varphi_i)$  and  $S_n$  satisfies a goal  $G$ . Let  $R_0$  be another state such that  $R_0 \models S_0$ , then there exists at least one applicable plan  $(O_1\psi_1 \dots O_n\psi_n)$  with  $R_i = apply(R_{i-1}, O_i, \psi_i)$  such that  $\varphi_i \subseteq \psi_i$  and  $R_n \models G$ .*

Using the previous result every  $R_i \models S_i$  with  $\varphi_i \subseteq \psi_i$ . Each  $O_i$  is applicable in  $R_i$  because it is applicable in  $S_i$  that is more generic. Finally, if  $R_n \models S_n$  then  $R_n \models G$ .

## 7 Related Work and Conclusions

In [16] a framework is presented for describing and analyzing systems where cases are reused at different levels of abstraction. Systems are classified according to different criteria: the kind of stored cases (abstract and concrete); how abstract cases are acquired (manually or automatically); whether abstract cases are used for indexing; whether abstract cases are directly reused or refined into concrete cases; and whether storing abstract cases allows to delete concrete ones. The work presented here can be classified within this framework as one where: only abstract cases are stored; abstract cases are automatically obtained; abstract cases are used for indexing; abstract cases are refined into concrete cases when reused; and only abstract cases are stored.

Several works have emphasized the integration of generative and case-based planning as a way of coping with those situations where a complete domain theory and/or a complete world state are not available [17,18]. In such mixed-initiative planners a conversational case-based planner takes control whenever none of the generative planner operators can be applied. A case-based planner can generate answers given incomplete domain theories, and its conversational component may gather additional information from the user about the world state so that the generative planner can later proceed. Without the conversational component, the work described in [19] extends HTN planning to cope with domains where there is no complete domain theory by using cases instead of methods for task decomposition. However, these systems reuse cases without adaptation (basically a case propose a task decomposition), and do not consider abstracting the cases. Also, the system can not demonstrate the correctness of its answers since no preconditions are represented for the cases, and it is the user responsibility to choose among retrieved cases which one is going to be applied.

Regarding the use of abstraction for indexing, the work described in [20,21] proposes the use of abstraction of the planning states as an indexing mechanism. For every plan recorded as a case in the system, its intermediate states are abstracted through a simple mechanism that replaces literals by its type ('PostOffice-1' is replaced by 'Location') and such abstractions serve as indexes for plans containing the original concrete states. This technique is used in the context of plan recognition, where at every state the system tries to identify the next most probable action based on recorded cases.

Regarding plan adaptation, the work described in [22] proposes a knowledge-based technique that can modify a retrieved plan so that whenever some goals of the retrieved plan are not required for the current situation, superfluous actions are eliminated. The technique is specially designed for real-time strategy games where plan generation and execution is interleaved because the post-conditions of the actions can not be guaranteed and no complete plan can be generated before-hand. They also propose an additional step of case-based planning for obtaining additional goals not fulfilled by the retrieved plan. The idea of re-planning fragments of a retrieved case is also presented in [23] where they apply an adaptation-guided similarity measure [24], based on the well known FF planning heuristic, to identify portions of a retrieved plan that can be improved by

re-planning. They actually follow a two step process, first repairing a retrieved plan with a generative planner (a STRIPS-like version of the original FF planning system) and then trying to improve the quality of the resulting plan by substituting portions of it with sub-plans retrieved from the case base. Nevertheless, such method has only been tested in STRIPS like simple problems and it is unclear how well it can scale-up to complex domains.

In this paper, we have presented a case-based planning approach for reusing plans obtained by a generative planner that represents the state of the world using Description Logics. Cases are automatically abstracted through the knowledge included in the domain ontology used to represent the world state in the planner. We are able to reuse available ontologies which have not been specifically designed for planning. We have also demonstrated that, under certain assumptions, when the goal of an abstract case is subsumed by the goal of a new problem, and the case precondition is satisfiable in the initial state of the new problem, according to the domain ontology used to abstract the cases, then the abstract case can be reused to obtain a correct solution for the new problem.

Regarding implementation, the generative planner has been actually deployed under the name of DLPlan<sup>3</sup>: a planner that takes as input a domain model represented in OWL-DL and employs a DLs engine (Pellet) to make inferences about the domain axioms during planning. DLPlan can solve both hierarchical and non-hierarchical planning problems. Our short term goal is to incorporate the case-based techniques here described into the system.

## References

1. Maximini, K., Maximini, R., Bergmann, R.: An Investigation of Generalized Cases. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 261–275. Springer, Heidelberg (2003)
2. Brachman, R., Levesque, H.: Knowledge Representation and Reasoning. Morgan Kaufmann Publishers, San Francisco (2004)
3. Gil, Y.: Description Logics and Planning. *AI Magazine* 26, 73–84 (2005)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, New York (2003)
5. W3C Consortium: OWL web ontology language guide. W3C recommendation (2004), <http://www.w3.org/tr/owl-guide/>
6. Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C.: *A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.0* (2004)
7. Sirin, E.: *Combining Description Logic Reasoning with AI Planning for Composition of Web Services*. PhD thesis, Department of Computer Science, University of Maryland (2006)
8. Fikes, R., Nilsson, N.J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artif. Intell.* 2, 189–208 (1971)
9. Finger, J.J.: *Exploiting Constraints in Design Synthesis*. PhD thesis, Stanford University, Stanford, CA, USA (1987)

<sup>3</sup> Freely available at <http://sourceforge.net/projects/dlplan/>

10. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating Description Logic ABoxes. In: KR, pp. 46–56 (2006)
11. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating Description Logics and Action Formalisms: First Results. In: AAAI, pp. 572–577 (2005)
12. Reiter, R.: On Specifying Database Updates. *J. Log. Program.* 25, 53–91 (1995)
13. Katsuno, H., Mendelzon, A.O.: On the Difference Between Updating a Knowledge Base and Revising it. In: Ganderfors, P. (ed.) *Cambridge Tracts in Theoretical Computer Science*, vol. 29, pp. 183–203. Cambridge University Press, Cambridge (1991)
14. Herzig, A., Rifi, O.: Propositional Belief Base Update and Minimal Change. *Artif. Intell.* 115, 107–138 (1999)
15. Ginsberg, M.L., Smith, D.E.: Reasoning About Action I: A Possible Worlds Approach. *Artif. Intell.* 35, 165–195 (1988)
16. Bergmann, R., Wilke, W.: On the Role of Abstraction in Case-Based Reasoning. In: Smith, I.F.C., Faltings, B. (eds.) *EWCBR 1996*. LNCS, vol. 1168, pp. 28–43. Springer, Heidelberg (1996)
17. Muñoz-Avila, H., Aha, D.W., Breslow, L., Nau, D.S., Weber-Lee, R.: Integrating Conversational Case Retrieval with generative Planning. In: Blanzieri, E., Portinale, L. (eds.) *EWCBR 2000*. LNCS, vol. 1898, pp. 210–221. Springer, Heidelberg (2000)
18. Muñoz-Avila, H., McFarlane, D.C., Aha, D.W., Breslow, L., Ballas, J.A., Nau, D.S.: Using Guidelines to Constrain Interactive Case-Based HTN Planning. In: Althoff, K.D., Bergmann, R., Branting, K. (eds.) *ICCBR 1999*. LNCS, vol. 1650, pp. 288–302. Springer, Heidelberg (1999)
19. Macedo, L., Cardoso, A.: Case-Based, Decision-Theoretic, HTN Planning. In: Funk, P., González-Calero, P.A. (eds.) *ECCBR 2004*. LNCS, vol. 3155, pp. 257–271. Springer, Heidelberg (2004)
20. Kerkez, B., Cox, M.T.: Incremental Case-Based Plan Recognition Using State Indices. In: Aha, D.W., Watson, I. (eds.) *ICCBR 2001*. LNCS, vol. 2080, pp. 291–305. Springer, Heidelberg (2001)
21. Kerkez, B., Cox, M.T.: Incremental Case-Based Plan Recognition with Local Predictions. *International Journal on Artificial Intelligence Tools* 12, 413–463 (2003)
22. Sugandh, N., Ontañón, S., Ram, A.: Real-Time Plan Adaptation for Case-Based Planning in Real-Time Strategy Games. In: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (eds.) *ECCBR 2008*. LNCS, vol. 5239, pp. 533–547. Springer, Heidelberg (2008)
23. Tonidandel, F., Rillo, M.: Case Adaptation by Segment Replanning for Case-Based Planning Systems. In: Muñoz-Avila, H., Ricci, F. (eds.) *ICCBR 2005*. LNCS, vol. 3620, pp. 579–594. Springer, Heidelberg (2005)
24. Tonidandel, F., Rillo, M.: An Accurate Adaptation-Guided Similarity Metric for Case-Based Planning. In: Aha, D.W., Watson, I. (eds.) *ICCBR 2001*. LNCS, vol. 2080, pp. 531–545. Springer, Heidelberg (2001)

# A Scalable Noise Reduction Technique for Large Case-Based Systems

Nicola Segata<sup>1</sup>, Enrico Blanzieri<sup>1</sup>, and Pádraig Cunningham<sup>2</sup>

<sup>1</sup> DISI, University of Trento, Italy

<sup>2</sup> Computer Science, University College Dublin, Dublin, Ireland

**Abstract.** Because case-based reasoning (CBR) is instance-based, it is vulnerable to noisy data. Other learning techniques such as support vector machines (SVMs) and decision trees have been developed to be noise-tolerant so a certain level of noise in the data can be condoned. By contrast, noisy data can have a big impact in CBR because inference is normally based on a small number of cases. So far, research on noise reduction has been based on a majority-rule strategy, cases that are out of line with their neighbors are removed. We depart from that strategy and use local SVMs to identify noisy cases. This is more powerful than a majority-rule strategy because it explicitly considers the decision boundary in the noise reduction process. In this paper we provide details on how such a local SVM strategy for noise reduction can be made scale to very large datasets ( $> 500,000$  training samples). The technique is evaluated on nine very large datasets and shows excellent performance when compared with alternative techniques.

## 1 Introduction

While many learning algorithms can be modified to be noise tolerant it is difficult to make instance-based learning (IBL) algorithms such as  $k$ -nearest neighbour ( $k$ -NN) classifiers or case-based reasoning (CBR) robust against noise. Thus noise reduction is important for improving generalisation accuracy in IBL. A further motivation for noise reduction in CBR is explanation – a capability that is perceived to be one of the advantages of CBR [1,2]. Since case-based explanation will invoke individual cases as part of the explanation process it is important that noisy cases can be eliminated if possible. Even if noise reduction will not improve the classification accuracy of learning algorithms that have been developed to be noise tolerant, researchers have argued that noise reduction as a preprocessing step can simplify resulting models, an objective that is desirable in many circumstances [3].

In  $k$ -NN and CBR the problem of noise reduction has traditionally been considered part of the larger problem of case-base maintenance. Since large training sets can influence the response time of lazy learners an extensive literature is dedicated to the development of data reduction techniques that preserve training set competence. On the other hand, there has also been a lot of research on competence enhancing techniques that preprocess the training data to remove



noisy instances. Much of this work is based on variations on the majority rule principle whereby examples that disagree with their neighbors are removed. In the work reported here we construct local support-vector machines (SVMs) in the region of a data point and use the decision surface of the SVM to determine if a data point should be removed. We present a strategy that reduces the number of local SVMs used in the noise reduction process to manageable proportions.

It is well-known that the classification error of the NN classifier is bounded by twice the Bayes error as the number of training samples  $n$  goes to infinity. This bound can be lowered to the Bayes error using the  $k$ -NN classifier with an high  $k$  (it is required that  $k \rightarrow \infty$ ,  $n \rightarrow \infty$ ,  $k/n \rightarrow 0$ ), or using the 1-NN classifier on an edited training set such that it guarantees the perfect training set classification [4]. This suggests that, for practical problems, if we are interested mainly in generalization accuracies, two aspects are crucial: the ability to detect and remove noisy samples in order to theoretically approach the Bayes error with the 1-NN classifier, and the possibility to use as much data as possible in order to approximate the  $n \rightarrow \infty$  condition. We tackled the first aspect with LSVM noise reduction (see [5] and Section 2.1) based on Local SVM [6,7] which was demonstrated to outperform existing noise reduction techniques mainly due by bringing local class boundaries into consideration. In this paper we focus on the second aspect, namely on developing a noise reduction technique (based on LSVM noise reduction) that can be applied to large and very large datasets. The developed noise reduction method, called FaLKNR, is part of the Fast Local Kernel Machine Library (FaLKM-lib) [8] freely available for research and education purposes at <http://disi.unitn.it/~segata/FaLKM-lib>.

## 1.1 Motivation

The local nature of case-based reasoning entails a vulnerability to noise in training data. Thus CBR has a dependency on individual training samples that other supervised learning techniques do not have. Other techniques have been developed to be noise tolerant by incorporating into the induction process mechanisms that attempt to avoid overfitting to noise in the training set. Examples of this include early stopping for artificial neural networks [9], the post-pruning of decision trees [10] and using soft-margin Support Vector Machines which relax the constraints on the margin maximisation [11]. However, instance based techniques such as  $k$ -NN that rely on specific retrieved instances for induction are affected by noise. These techniques generally lack the induction step that other noise tolerant techniques can adapt. The dependance on the specific retrieved instances can be reduced by retrieving more instances (i.e.  $k$ -NN, with  $k > 1$  is more noise tolerant than 1-NN) but accuracy will not always increase with larger values of  $k$ . At some point a large  $k$  will result in a neighbourhood that crosses the decision surface and accuracy will drop.

An additional motivation for noise reduction in IBL associated with this dependency on individual training samples is case-based explanation. A learning system that can provide good explanations for its predictions can increase user confidence and trust and give the user a sense of control over the system [12].

Case-based explanations are generally based on a strategy of presenting similar past examples to support and justify the predictions made [2,13]. If specific cases are to be invoked as explanations then noisy cases need to be identified and removed from the case-base.

Despite the importance of noise reduction for IBL and CBR, little work has been done on making competence enhancement techniques applicable to large data collections in order to better approach the theoretical Bayes error rate on unseen samples with the simple NN classifier.

Finally there are specific application areas where noise reduction is important. It is generally accepted that inductive learning systems in the medical domain are dependent on the quality of the data [14] and there has been significant research into data cleansing in bioinformatics [15,16,3,17]. Although instance based techniques such as  $k$ -NN are not generally used for classification in much of this research, noise reduction is an important element in the process as it can result in the simplification of the models created. Lorena and Carvalho [3], for example, found that preprocessing the training data to remove noise resulted in simplifications in induced SVM classifiers and higher comprehensiveness in induced decision tree classifiers. Both in data cleansing in bioinformatics and as a preprocessing step for different classifiers, the scalability issue is often crucial.

## 1.2 Related Work

Editing strategies for IBL and CBR may have different objectives as discussed for example by Wilson and Martinez [18] and Brighton and Mellish [19]. According to them, editing techniques can be categorised as competence preservation or competence enhancement techniques. Competence preservation techniques aim to reduce the size of the training set as much as possible without significantly affecting the generalisation accuracy thus achieving a reduction in the storage requirements and increasing the speed of execution. The main goal of competence enhancement techniques is to increase the generalisation accuracy primarily by removing noisy or corrupt training examples. Some strategies aim to tackle both objectives at the same time and for this reason are called hybrid techniques [19].

In this work we are focusing on competence enhancement and for this reason we briefly review the existing techniques that demonstrate good generalisation accuracy (for a recent review of competence preservation and hybrid methods the reader can refer to [5]).

**Competence Enhancement Methods.** The objective of competence enhancement methods is to remove noisy, mislabelled and borderline examples that are likely to cause misclassification thus allowing  $k$ -NN classifiers to model smoother decision surfaces. They start with Wilson's *Edited Nearest Neighbor* algorithm (ENN) [20], a decremental strategy that simply removes from the training set those examples that do not agree with the majority of their  $k$  nearest neighbours.

Tomek [21] proposed two improvements to ENN; *Repeated Edited Nearest Neighbor* (RENN) and *All- $k$ NN* (AkNN). Both RENN and AkNN make multiple passes over the training set repeating ENN. RENN just repeats the ENN

algorithm until no further eliminations can be made from the edited set while AkNN repeats ENN for each sample using incrementing values of  $k$  each time and removing the sample if its label is not the predominant one at least for one value of  $k$ . It is worth noting that for  $k = 1$ , ENN and AkNN are equivalent and for  $k > 1$  AkNN is more aggressive than ENN.

A slightly different approach is introduced by Koplowitz and Brown [22] which considers the relabelling of some examples instead of their removal. This idea is expanded on by Jiang and Zhou [23] using an ensemble of neural networks to determine the label for the examples that are to be relabelled. Another modification of ENN and RENN proposed by Sánchez et al [24] entails substituting the  $k$  nearest neighbours with the  $k$  nearest centroid neighbours ( $k$ -NCN) where the neighbourhood of an example is defined not only based on distances from the example but also on the symmetrical distribution of examples around it.

The detecting of mislabeled samples in high-dimensional spaces with small sample size (typical of high-throughput bioinformatics) is addressed by Malossini et al [15] based on a leave-one-out perturbation matrix and a measure of the stability of the label of a sample with respect to label changes of other samples.

In the context of editing training data for spam filtering, Delany and Cunningham [25] advocate putting the emphasis on examples that cause misclassifications rather than the examples that are themselves misclassified. The method which is called *Blame Based Noise Reduction* (BBNR) enhances the competence properties of *coverage* and *reachability* with the concept of a *liability set*.

**Benchmarking Noise Reduction and Generalisation Accuracy Enhancement.** The main editing techniques developed before 2000 have been extensively evaluated by Wilson and Martinez [18]. The overall result of their analysis is that DROP3 (a hybrid technique [18]) has the best mix of generalisation accuracy and storage reduction. However, looking at generalisation capability only, they conclude that their DROP3 method has somewhat lower accuracy than the group of methods including ENN, RENN and AkNN. In particular, among these last three methods, AkNN has “the highest accuracy and lowest storage requirements in the presence of noise” [18]. The comparisons of ICF (a competence preservation method [19]) with DROP3 by Brighton and Mellish [19] highlights that they have similar performance but, considering the accuracy results only, it is clear that ENN outperforms both in the majority of the datasets.

$k$ -NCN seems to be more accurate than AkNN and ENN [24], but the analysis is performed on five datasets only and does not include an assessment of statistical significance. Moreover  $k$ -NCN substitutes real samples with synthetic ones preventing case-based explanation. Without considering the competence preserving methods as our objective is competence enhancement, the remaining approaches (including the neural network ensemble approach presented by [23] and KGCM [26]) do not provide any comparison with ENN, RENN or AkNN and the reproduction of these techniques is non trivial as they are embedded in complex frameworks. The approach proposed by Malossini et al. [15] is conceived for very high dimensional datasets with very few samples and thus it is not suitable for general real datasets.

Taking this into consideration, we chose to empirically compare our proposed noise reduction technique with AkNN as, despite its simplicity, it still represents the state-of-the-art for competence enhancement. We also include comparisons with ENN and RENN as they are the most popular noise reduction techniques used in the literature.

As far as we know, the only work that focus on the computational performances of the editing techniques, is the competence preservation method presented by Angiulli [27] where processing time is  $O(n^2)$  in the number of examples.

## 2 Fast Noise Reduction with Local Kernel Machines

We introduce here the **F**ast **L**ocal **K**ernel Machine **N**oise **R**eduction (FaLKNR), which is scalable for large datasets. It is developed starting from the LSVM noise reduction method, described in [5] and summarized in Section 2.1. Various modifications and optimization strategies are introduced in Section 2.2 to make it suitable for large CBR systems. In Section 2.3 the computational complexity of the obtained noise reduction technique is analysed.

### 2.1 The LSVM Noise Reduction Technique

The local SVM noise reduction [5] is a reduction technique based on local Support Vector Machines [6,7] (LSVM) which brings the benefits of maximal margin classifiers to bear on noise reduction. This provides a more robust alternative to the majority rule used by most competence enhancing techniques augmenting it with the kernel-space maximal margin principle. Roughly speaking, LSVM trains for each training sample an SVM on its neighbourhood and if the SVM classification for the central sample disagrees with its actual class there is evidence in favour of removing it from the training set. By extending LSVM with a probabilistic output it is possible to apply it on the training set to remove noisy, corrupted and mislabelled samples. In other words local SVM noise reduction removes those samples that, with respect to the maximal separating hyperplanes built on the feature space projections of their neighbourhoods, are too close to or on the wrong side of the decision boundary.

We need to briefly recall some SVMs [11] basics. Consider a training set  $X$  with samples  $(x_i, y_i)$  with  $i = 1, \dots, n$ ,  $x_i \in \mathbb{R}^p$  and  $y_i \in \{+1, -1\}$ . The SVM decision rule is  $\text{SVM}(x) = \text{sign}(\langle w, \Phi(x) \rangle_{\mathcal{F}} + b)$  where  $\Phi(x) : \mathbb{R}^p \rightarrow \mathcal{F}$  is a mapping in a transformed feature space  $\mathcal{F}$  with inner product  $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ . The empirical risk term is controlled through the following set of constraints:

$$y_i (\langle w, \Phi(x_i) \rangle_{\mathcal{F}} + b) \geq 1 - \xi_i \quad \xi_i \geq 0, i = 1, \dots, n \quad (1)$$

where the slack variables  $\xi_i$  allow some misclassification on the training set. The problem can be reformulated with Lagrange multipliers  $\alpha_i$  ( $i = 1, \dots, n$ ) and introducing a positive definite kernel (PD) function [1]  $K(\cdot, \cdot)$  that substitutes

<sup>1</sup> For convention we refer to kernel functions with the capital letter  $K$  and to the number of nearest neighbours with the lower-case letter  $k$ .

the scalar product in the feature space  $\langle \Phi(x_i), \Phi(x) \rangle_{\mathcal{F}}$  obtaining the following decision rule:  $SVM(x) = sign(\sum_{i=1}^n \alpha_i y_i K(x_i, x) + b)$ .

In the case of *local SVM*, that belongs to the class of local learning algorithms [28,29], the classification problem is tackled differently from SVM. In fact, instead of estimating a global decision function with a low probability of errors on *all* possible unseen samples, local SVM tries to estimate a decision function with a low probability of error on labeling a *given* point. This is achieved by retrieving for each point a neighborhood on which an SVM is built subject to the following constraints:

$$y_{r_x(i)} (w \cdot \Phi(x_{r_x(i)}) + b) \geq 1 - \xi_{r_x(i)}, \text{ with } i = 1, \dots, k$$

where  $r_{x'} : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$  is a function that reorders the indexes of the training points defined as:

$$\begin{cases} r_{x'}(1) = \operatorname{argmin}_{i=1, \dots, n} \|\Phi(x_i) - \Phi(x')\|^2 \\ r_{x'}(j) = \operatorname{argmin}_{i=1, \dots, n} \|\Phi(x_i) - \Phi(x')\|^2 \quad i \neq r_{x'}(1), \dots, r_{x'}(j-1) \text{ for } j = 2, \dots, n \end{cases} \quad (2)$$

The computation is expressed in terms of kernels as  $\|\Phi(x) - \Phi(x')\|^2 = \langle \Phi(x), \Phi(x) \rangle_{\mathcal{F}} + \langle \Phi(x'), \Phi(x') \rangle_{\mathcal{F}} - 2 \cdot \langle \Phi(x), \Phi(x') \rangle_{\mathcal{F}} = K(x, x) + K(x', x') - 2 \cdot K(x, x')$ . In this way the decision rule for a sample  $x$  and a training set  $X$  is:

$$kNNSVM(x; X) = sign \left( \sum_{i=1}^k \alpha_{r_x(i)} y_{r_x(i)} K(x_{r_x(i)}, x) + b \right). \quad (3)$$

The probability output for this method can be obtained (following the approach of Platt [30] refined by Lin et al [31]) as:

$$\hat{p}^{kNNSVM}(y = +1|x; X) = [1 + \exp(A \cdot kNNSVM(x; X) + B)]^{-1}$$

Noisy training samples can be detected by applying the LSVM method on each training sample. The edited training set  $X' \subseteq X$  of training samples without the noisy samples can be thus defined as

$$X' = \{x_i \in X \mid \hat{p}^{kNNSVM}(y = y_i|x_i; X \setminus \{x_i\}) > \gamma\}. \quad (4)$$

where  $\gamma$  is a threshold that can be manually tuned to modify the amount of noise to be removed and the probability level associated with non-noisy samples. Notice that, for each  $i$ , the training sample  $x_i$  that is tested to check if it is noisy or not, is excluded from the training set on which the local SVM is trained. This is done so that the testing point is not included in the SVM training process.

The experiments carried out in [5] highlight that LSVM noise reduction overcomes the state-of-the-art noise reduction techniques for IBL in a number of real datasets (with statistical significance), for datasets affected by Gaussian noise and in the presence of uneven class densities.

## 2.2 Making LSVM Noise Reduction Scalable for Large Datasets

Our objective here is to reduce the computational overhead of LSVM noise reduction. As a first step, we need to conceptually separate the point for which a local SVM model is trained and the local SVM model itself in order to let multiple points be evaluated with the same model thus reducing the number of models that need to be retrieved and built. For achieving this we need to generalize the decision rule of  $k$ NNSVM considering explicitly the training point  $t$  which acts as the center of the model:

$$k\text{NNSVM}_t(x; X) = \text{sign} \left( \sum_{i=1}^k \alpha_{r_t(i)} y_{r_t(i)} K(x_{r_t(i)}, x) + b \right)$$

where  $\alpha_{r_t(i)}$  and  $b$  come from the training of an SVM on the  $k$ -nearest neighbors of  $t$  in the feature space and  $r_t(i)$  is the ordering function defined above.

The problem which arises now consists in covering the entire training set with a relatively small number of local SVM models and to assign a certain number of points to be classified as noise or not to each model. This is done by assigning to the local model centered in a point  $c$  not only  $c$  itself but also the first  $k'$  (with  $k' \leq k$ ) nearest neighbors of  $c$ . In this way we aim to make a compromise (controlled by  $k'$ ) between the number of local SVMs trained and the specificity of the local models to the data points being assessed. The set of points used to select the  $k$ -nearest neighbors for the models can be defined as follows.

**Definition 1.** Given  $k' \in \mathbb{N}$ , a  $k'$ -neighborhood covering set of centers  $\mathcal{C}_{k'} \subseteq X$  is a subset of the training set such that the following holds:

$$\bigcup_{c \in \mathcal{C}_{k'}} \{x_{r_c(i)} \mid i = 1, \dots, k'\} = X.$$

Definition 1 means that the union of the sets of the  $k'$ -nearest neighbors of  $\mathcal{C}_{k'}$  corresponds to the whole training set. Theoretically, for a fixed  $k'$ , the minimization of the number of local SVMs that we need to train can be obtained computing the SVMs centered on the points contained in the *minimal*  $k'$ -neighborhood covering set of centers<sup>2</sup>  $\mathcal{C}$ . However, since identifying the minimal  $\mathcal{C}$  is not a simple and computationally easy task, a practical strategy to select each  $c_i \in \mathcal{C}$  is the following:

$$c_i = x_j \in X \quad \text{with } j = \min (l \in \{1, \dots, n\} \mid x_l \in X \setminus X_{c_i})$$

$$\text{where } X_{c_i} = \bigcup_{l < i} \{x_{r_{c_l}(h)} \mid h = 1, \dots, k'\}. \quad (5)$$

The idea of this definition is to recursively take as centers those points which are not  $k'$ -neighbors of any point that has already been taken as center. So  $c_1 = x_1$  corresponds to the first point of  $X$  since, being  $c_1$  the first center,

<sup>2</sup> From now on we simply denote  $\mathcal{C}_{k'}$  with  $\mathcal{C}$  because we do not discuss here particular values for  $k'$ .

the union of the neighbors of the other centers is empty;  $c_2$ , instead, is the point with the minimum index taken from the set obtained eliminating from  $X$  all the  $k'$ -neighbors of  $c_1$ . The procedure is repeated until all the training points are removed from  $X$  where  $X$  is a random reordering of the training set. The data is randomized in order to avoid the possibility that a training set in which the points are inserted with a particular spatial strategy affects the spatial distribution of the  $k'$ -neighborhood covering centers.

The reason why we adopt this non-standard clustering method is twofold: from one side we want each cluster to contain exactly  $k$  samples in order to be able to derive rigorous complexity bounds, from the other side in this way we are able to select a variable number of samples that are in the central region (at least from a neighborhood viewpoint) of each cluster. Moreover the proposed clustering strategy follows quite naturally from the  $k$ NNSVM approach.

With this setting, we need to train only  $|\mathcal{C}|$  SVMs centered on each  $c \in \mathcal{C}$  obtaining the following models:

$$k\text{NNSVM}_c(x; X), \quad \forall c \in \mathcal{C}.$$

Now we have to link the points of the training set with the precomputed SVM models. This is necessary because a point can lie in the  $k'$ -neighborhood of more than one center. In particular we want to consider the assignments of each training point to a unique model such that it is in the  $k'$ -neighborhood of the center on which the model is built. Formally this is done with the function  $\text{cnt}(t) : X \rightarrow \mathcal{C}$  that assigns each point in the training set to a center:

$$\begin{aligned} \text{cnt}(x_i) = x_j \in \mathcal{C} \quad & \text{with } j = \min \left\{ l \in \{1, \dots, n\} \mid x_l \in \mathcal{C} \text{ and } x_i \in X_{x_l} \right\} \\ & \text{where } X_{x_l} = \left\{ x_{r_{x_l}(h)} \mid h = 1, \dots, k' \right\}. \end{aligned} \tag{6}$$

With the  $\text{cnt}$  function, each training point is assigned to the first center whose  $k'$ -nearest neighbors set includes the training point itself. The order of the  $c_i$  points derives from the randomization of  $X$  used for defining  $\mathcal{C}$ . In this way each training point is univocally assigned to a center and so the decision function of each training set sample, is simply:

$$\text{FastLSVM}(x; X) = k\text{NNSVM}_c(x; X) \quad \text{with } c = \text{cnt}(x_{r_x(1)}) \tag{7}$$

The edited training set  $X' \subseteq X$  without the noisy samples becomes

$$X' = \{x_i \in X \mid \text{FastLSVM}(x; X) = y_i\}.$$

Notice that we do not convert the output of the SVM to a probability because we want to avoid the tuning of the threshold parameter  $\gamma$  of Equation 4 (in the case of datasets with two classes it is conceptually equivalent to set the threshold to 0.5) and to make the SVM model construction faster (the probabilistic approach following [30] and [31] requires a cross-validation step).

**Adopting the Cover Tree Data structure.** A Cover Tree is a data structure introduced in [32] for performing fast and efficient non-approximated nearest neighbor operations. Cover Trees can be applied in general metric spaces without assumptions on the structure and thus also in Hilbert Spaces calculating the distances by means of kernel functions using the kernel trick.

In more detail, a Cover Tree can be viewed as a subgraph of a navigating net [33] and it is a leveled tree in which each level (indexed by a decreasing integer  $i$ ) is a cover (i.e. is representative) for the level beneath it. Every node of a Cover Tree  $T$  is associated with a point of a dataset  $S$ . Denoting with  $C_i$  the set of points associated with nodes in  $T$  at level  $i$ , with  $base > 1$  a constant, and with  $dist(\cdot, \cdot)$  the distance function defining the metric of the space, the invariants of a Cover Tree are:

**Nesting.**  $C_i \subset C_{i-1}$

**Covering tree.** For every  $p \in C_{i-1}$  there exists a  $q \in C_i$  such that  $dist(p, q) < base^i$  and the node in level  $i$  associated with  $q$  is a parent of the node in level  $i - 1$  associated with  $p$ .

**Separation.** For all distinct  $p, q \in C_i$ ,  $dist(p, q) > base^i$ .

Intuitively, nesting means that once a point appears in a level, it is present for every lower level. A covering tree implies that every node has a parent in the higher level such that the distance between the respective points is less than  $base^i$ , while separation assures that the distance between every pair of points associated to the nodes of a level  $i$  is higher than  $base^i$ .

Cover Trees have state-of-the-art performance for exact nearest neighbor operations for general metrics in low-dimensional spaces both in terms of computational complexity and space requirements. As theoretically proved by Beygelzimer et al. [32], the space required by the Cover Tree data-structure is linear in the dataset size ( $\mathcal{O}(n)$ ) while the computational time of single point insertion, deletion and exact nearest neighbor query is logarithmic ( $\mathcal{O}(\log n)$ ). For these reasons we used Cover Trees for implementing FaLKNR (and also ENN, RENN and AkNN).

Moreover we exploit the separation invariant for selecting the centers of the local models. More precisely, the idea is to apply Equation 5 used to chose the centers for FastLSVM not to a random reordering of the training set, but to the array of points  $X_{CT}$  obtained traversing the Cover Tree built on the training set from the root to the lowest level exploring each level completely before exploring the lower one. In this way, the separation invariant ensures that the distances of subsequent samples in  $X_{CT}$  tends to be maximized; this causes the centers, and thus the local SVM models, to be more evenly distributed in the datasets. From another viewpoint, choosing iteratively the centers as far as possible, minimizes the situations in which some of the  $k'$ -nearest neighbors of the two centers coincides thus leading to a lower number of models needed to cover the entire dataset. Preliminary experiments demonstrated that the use of this reordering strategy decreases the number of local model that needs to be built by between 5% and 10%.



**Local Model Selection for FaLKNR.** When training an SVM model, it is crucial to choose a proper kernel, to carefully tune the kernel parameters and to set the soft margin regularization constant  $C$ . One of the most effective and practical approaches for doing this is based on cross-validation. However, for FaLKNR, cross validation is not a suitable technique because of its computational overhead, and for the fact that the parameters are estimated globally, whereas FaLKNR has the opportunity to set different values for each local model.

In our experiments we will use the RBF kernel which is a general purpose kernel that has demonstrated very high classification accuracies for SVM. Its definition is  $k^{rbf}(x, x') = \exp\left(-\frac{\|x-x'\|^2}{\sigma}\right)$ , where  $\sigma$  is the width of the kernel. In our approach we set  $\sigma$  to be the double of the squared median of the histogram of the distances in the local model. More formally,  $\sigma = 2 \cdot m^2[\|x - x'\|_{\mathbb{R}^p_k}]$  where  $m[\|x - x'\|_{\mathbb{R}^p_k}]$  is the median of the distance distribution in  $\mathbb{R}^p$  of the  $k$  points of the local model. This procedure is motivated by the fact that the obtained  $\sigma$  value is of the same order of magnitude as the distances that it weights. In this way the kernel width is adaptive to the possibly different characteristics of different sub-regions of the training set. For non-low values of  $k$   $\sigma$  is computed on a random subset of points for computational reasons.

The regularization parameter  $C$  is chosen with local model selection on a subset of the local SVM models (typically 10 models). Since the local models are used to classify only the  $k'$  most internal points, they must be tuned to be predictive especially in this internal region. For this reason we modified the standard  $\kappa$ -fold cross-validation approach in the following way:

1. we separate the  $k'$  most internal samples, called  $S$ , from the remaining external points, called  $S^E$ ;
2. we randomly split  $S$  in  $\kappa$  disjoint internal validation sets  $S_i$  with  $0 < i < \kappa$ ;
3. for each fold  $i$  we train a model with the  $S^E \cup (S \setminus S_i)$  set evaluating it on the correspondent  $S_i$  set, taking the mean of the accuracies on the  $\kappa$  folds.

The set of possible  $C$  values from which the best parameter is estimated with this modified  $\kappa$ -fold cross-validation is  $\{1, 10, 100\}$  using  $\kappa = 10$ .

### 2.3 Computational Complexity of FaLKNR

Hypothesizing the worst scaling behaviour for the training of each local SVM model to be  $\mathcal{O}(k^3)$ , and remembering that the nearest neighbor operations with Cover Trees can be done in  $\log(n)$ , FaLKNR requires  $\mathcal{O}(n \log n)$  for building the Cover Tree,  $\mathcal{O}(|\mathcal{C}| \cdot \log n \cdot k)$  for retrieving the local models,  $\mathcal{O}(|\mathcal{C}| \cdot k^3)$  for training the local SVMs, and  $\mathcal{O}(k \cdot n)$  for predicting if each training point is a noisy point or not. This means that the overall complexity of FaLKNR is  $\mathcal{O}(n \log n + |\mathcal{C}| \cdot \log n \cdot k + |\mathcal{C}| \cdot k^3 + k \cdot n)$ , which is, assuming a fixed and reasonably low value for  $k$ , sub-quadratic (in particular  $\mathcal{O}(n \log(n))$ ) even considering the worst case in which  $k' = 1$  and thus  $|\mathcal{C}| = n$ . Moreover, FaLKNR can be very easily parallelized, because the training (and testing) of the local SVMs can occur in parallel on different processors.

LSVM noise reduction, as presented in [5], has a complexity of  $\mathcal{O}(n^2 \log n + n \cdot k^3)$ . The only work that, as far as we know, is focused on computational

performances for noise reduction ([27]) has a complexity of  $\mathcal{O}(n^2)$ . ENN, using a brute-force nearest neighbor approach, scales like  $\mathcal{O}(n^2 \log k)$  but, using Cover Trees, its complexity can be lowered to  $\mathcal{O}(n \log n + k \cdot n \log n)$ , which is thus of the same complexity class of FaLKNR with respect to  $n$ . RENN and AkNN have the same complexity as ENN, with the addition of a small constant (for RENN the number of recursive applications, for AkNN the neighborhood size  $k$ ).

As for the computational space requirements, since FaLKNR performs SVM training on small subregions (assuming a reasonable low  $k$ ), there are no problems with fitting the kernel matrix into main memory. This results in an overall space requirement of  $\mathcal{O}(n + |\mathcal{C}| \cdot k^2)$ , i.e. linear in  $n$ .

### 3 Empirical Evaluation of FaLKNR

We compare FaLKNR to ENN, RENN and AkNN the state-of-the-art methods for competence enhancing as discussed in Section 1.2. The comparison is made on the basis of nearest neighbor (NN) generalisation accuracies. We implemented FaLKNR using our Cover Trees implementation and LibSVM [34] for local SVM training and prediction; the source code of FaLKNR is freely available as a module of the Fast Local Kernel Machine Library (FaLKM-lib) [8]. The Cover Trees are used to implement ENN and AkNN as well. Although it is not computationally efficient, RENN can be realised by simply recursively applying ENN until no samples are removed. LSVM noise reduction is not considered because is not scalable for large datasets<sup>3</sup>. The experiments are carried out on an AMD Athlon™ 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM.

#### 3.1 Experimental Procedure

The  $k$  and  $k'$  parameters of FaLKNR are set to 1000 and 250 respectively. There are no particular strategies to select such values, but we intuitively considered them a good compromise between local and global behaviours (for  $k$ ) and between generalisation accuracies and computational performance (for  $k'$ ). The other parameters are chosen or estimated as detailed in Section 2.2. In the case of ENN, RENN and AkNN we fixed  $k = 3$  as done, among others, by Wilson and Martinez [18]. Notice that, choosing an odd number for  $k$ , ties in the majority rule are avoided<sup>4</sup>. However, for AkNN, the  $k = 2$  case is considered and thus the number of ties in the majority rule can be large. Two versions of AkNN are thus taken into account: in AkNN a sample is removed in the case of a tie, while in AkNNc (more conservative) the sample is not removed in the case of a tie.

For the evaluation we used the datasets with less than 60 features and more than 45000 training samples available on the LibSVM [34] and UCI [35]

<sup>3</sup> LSVM noise reduction on the smallest dataset we present here takes more than 10 hours without considering model selection.

<sup>4</sup> Ties in the majority rule can still happen even with  $k = 3$  if multiple points are at the same distance from the query point at the  $k$ -th position. However in the datasets considered here the number of points at the same position is negligible and the dimensionality is low, and thus ties with odd  $k$  values are extremely rare.

**Table 1.** The datasets used for the empirical evaluation

name	# training samples	# testing samples	# features	# classes	source
ijcnn1	49990	91701	22	2	LibSVM Rep. <a href="#">34</a>
connect-4	50669	16888	41	3	UCI Rep. <a href="#">35</a>
seismic	78823	19705	50	3	LibSVM Rep. <a href="#">34</a>
acoustic	78823	19705	50	3	LibSVM Rep. <a href="#">34</a>
2-spirals	100000	100000	2	2	Segata et al. <a href="#">36</a>
census-income	199523	99762	41	2	UCI Rep. <a href="#">35</a>
poker-hand	300000	725010	10	2	UCI Rep. <a href="#">35</a>
rna	364651	121549	8	2	Uzilov et al. <a href="#">37</a>
cover-type	571012	10000	54	2	LibSVM Rep. <a href="#">34</a>

repositories, an artificial dataset described in [36](#) and the bioinformatics dataset provided in [37](#). If no separate testing sets are available we randomly chose one quarter of the data for testing, apart for the cover-type dataset for which we selected 10000 testing points (this because for this dataset it is necessary to have almost all the points for good classification results) and for poker-hand for which we added 275000 testing samples to the training set in order to make it larger. The datasets are listed in Table [1](#) and are all scaled in the range  $[0, 1]$  (apart for 2-spirals which is in the  $[-2, 2]$  range).

### 3.2 Results and Discussion

Table [2](#) reports the NN generalisation accuracies obtained using the original (unedited) training set and the training sets edited with the analysed techniques. FaLKNR improves on the accuracy achieved with the unedited training sets for 7 of the 9 datasets and in a number of cases the improvements are considerable. ENN, RENN, AkNN and AkNNc are also able to improve the NN generalisation accuracy in the majority of the datasets, but their improvements are *always* lower than the LSVM noise reduction ones. If we use the Wilcoxon signed-ranks test to assess the significance of this table of results [38](#), the improvements due to FaLKNR are statistically significant ( $\alpha = 0.05$ ) with respect to all the other analysed techniques and with respect to the unedited training set.

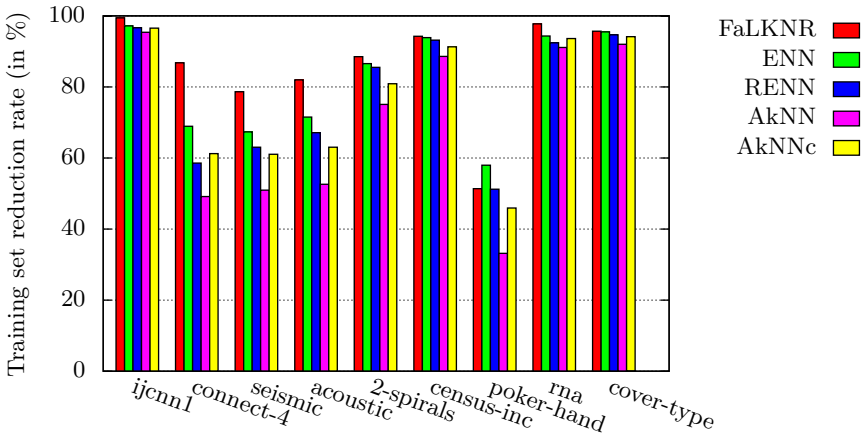
The total computational times for FaLKNR (including local model selection and the local SVM training/prediction) are between 39 seconds for ijcnn1 and about 38 minutes for poker-hand (2230 seconds). In the last column of Table [2](#) we report the speedups of FaLKNR with respect to ENN (implemented using Cover Trees). We chose ENN for this comparison because it is in any case faster than RENN and AkNN, and thus the speedups of FaLKNR with respect to RENN and AkNN are higher than reported in the table. The speedups are always higher than 1 except for the 2-spirals dataset (97 seconds for FaLKNR, 44 for ENN). These favourable computational results are due to the fact that it is faster to perform  $|\mathcal{C}|$  retrievals of the  $k = 1000$  nearest neighbors than it is to perform  $n$  retrievals of  $k = 3$  nearest neighbors. This advantage is maintained when training  $|\mathcal{C}|$  local SVMs, confirming that the training (and the prediction) of SVMs with 1000 points is extremely fast. The only dataset in which this does not hold is the

**Table 2.** NN accuracies using the analysed techniques to edit the training sets. In bold and italics are highlighted the best and worst results. We report also the speedups of FaLKNNR with respect to ENN (the fastest among ENN, RENN and AkNN).

dataset	NN accuracies (in %)						comput. speedup FaLKNNR vs ENN
	unedited	FaLKNNR	ENN	RENN	AkNN	AkNNc	
ijcnn1	96.6	<b>96.7</b>	96.3	<i>96.0</i>	<i>96.0</i>	96.2	1.6
connect-4	<i>66.2</i>	<b>69.8</b>	69.3	68.3	69.3	69.4	2.7
seismic	<i>65.3</i>	<b>73.3</b>	71.9	72.6	72.2	71.8	3.2
acoustic	<i>67.4</i>	<b>75.3</b>	73.7	74.2	74.0	73.8	8.0
2-spirals	<i>83.2</i>	<b>88.6</b>	87.6	88.1	87.9	87.7	0.5
census-inc	<i>92.6</i>	<b>94.5</b>	94.2	94.3	94.4	94.3	9.0
poker-hand	<i>56.6</i>	<b>60.7</b>	57.8	58.3	58.3	58.0	7.6
rna	<b>96.3</b>	95.8	<i>94.0</i>	<i>94.0</i>	94.3	94.3	6.1
cover-type	<b>95.8</b>	95.4	95.2	<i>95.0</i>	95.1	95.2	1.5

2-spiral dataset because it is a very complex classification problem and thus the local SVM models are rather slow to train and because it has only two features and thus the nearest neighbor operations of ENN are very efficient.

Although our objective here is competence enhancement, it is interesting to look at the size of the edited training sets reported in Fig. 1. The correlation between the unedited training set NN accuracies and the size of the edited training sets is evident, and this is an indirect confirmation that the tested techniques do home in on noisy samples. FaLKNNR is the method that removes less samples in almost all the datasets. One may thus argue that the reason why FaLKNNR outperforms the other techniques in improving NN accuracies is related to the fact that it is less aggressive in removing samples. However, we can notice that the difference in training set reduction between AkNN and AkNNc is consistent, but AkNNc does not permit better NN accuracies. This let us conclude that the advantages of FaLKNNR over other techniques is not simply due to its more conservative policy.



**Fig. 1.** Percentage sizes of the training sets edited with the analysed techniques

## 4 Conclusions

We have presented FaLKNR, a scalable noise reduction technique based on the predictions of a set of local SVM models built on the training set. FaLKNR is based on the LSVM noise reduction technique we presented in [5] and includes a number of optimizations to achieve a theoretical complexity bound of  $\mathcal{O}(n \log(n))$  for non high-dimensional data. This makes it possible to apply the method on datasets with more than 500000 samples. Our empirical evaluation carried out in comparison with the state-of-the-art noise reduction techniques represented by ENN, AkNN and RENN, demonstrated that FaLKNR is the fastest and permits the highest NN accuracy improvements.

## References

1. Leake, D.B.: CBR in context: The present and future. In: Leake (ed.) *Case Based Reasoning: Experiences, Lessons, and Future Directions*, pp. 3–30. MIT Press, Cambridge (1996)
2. Cunningham, P., Doyle, D., Loughrey, J.: An evaluation of the usefulness of case-based explanation. In: Ashley, K.D., Bridge, D.G. (eds.) *ICCBR 2003*. LNCS, vol. 2689, pp. 122–130. Springer, Heidelberg (2003)
3. Lorena, A.C., Carvalho, A.: Evaluation of noise reduction techniques in the splice junction recognition problem. *Genet. Mol. Biol.* 27, 665–672 (2004)
4. Devijver, P., Kittler, J.: *Pattern recognition: a statistical approach*, Englewood Cliffs, London (1982)
5. Segata, N., Blanzieri, E., Delany, S., Cunningham, P.: Noise reduction for instance-based learning with a local maximal margin approach. Technical Report DISI-08-056, DISI, University of Trento, Italy (2008)
6. Blanzieri, E., Melgani, F.: Nearest neighbor classification of remote sensing images with the maximal margin principle. *IEEE Trans. Geosci. Remote Sens.* 46(6) (2008)
7. Segata, N., Blanzieri, E.: Empirical assessment of classification accuracy of Local SVM. In: *Proc. of Benelearn*, pp. 47–55 (2009)
8. Segata, N.: FaLKM-lib v1.0: a Library for Fast Local Kernel Machines. Technical report, DISI, University of Trento, Italy (2009), <http://disi.unitn.it/~segata/FaLKM-lib>
9. Cataltepe, Z., Abu-mostafa, Y.S., Magdon-ismail, M.: No free lunch for early stopping. *Neural Comput.* 11, 995–1009 (1999)
10. Quinlan, J.: The effect of noise on concept learning. In: Michalski, R., Carbonell, J., Mitchell, T. (eds.) *Mach Learn.* Morgan Kaufmann, San Francisco (1986)
11. Cortes, C., Vapnik, V.: Support-vector networks. *Mach Learn.*, 273–297 (1995)
12. Roth-Berghofer, T.: Explanations and case-based reasoning: Foundational issues. In: Funk, P., González-Calero, P. (eds.) *ECCBR 2004*. LNCS, vol. 3155, pp. 389–403. Springer, Heidelberg (2004)
13. Nugent, C., Doyle, D., Cunningham, P.: Gaining insight through case-based explanation. *Int. J. Intell. Inf. Syst.* (2008)
14. Pechenizkiy, M., Tsymbal, A., Puuronen, S., Pechenizkiy, O.: Class noise and supervised learning in medical domains: The effect of feature extraction. In: *CBMS 2006*, Washington, DC, USA, pp. 708–713. IEEE Computer Society, Los Alamitos (2006)
15. Malossini, A., Blanzieri, E., Ng, R.T.: Detecting potential labeling errors in microarrays by data perturbation. *Bioinformatics* 22(17), 2114–2121 (2006)

16. Gamberger, A., Lavrac, N., Dzeroski, S.: Noise detection and elimination in data preprocessing: experiments in medical domains. *Appl. Artif. Intell.*, 205–223 (2000)
17. Tang, S., Chen, S.P.: Data cleansing based on mathematic morphology. In: *iCBBE 2008*, pp. 755–758 (2008)
18. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. *Mach. Learn.* 38(3), 257–286 (2000)
19. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discovery* 6(2), 153–172 (2002)
20. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Syst. Man Cybern.* 2(3), 408–421 (1972)
21. Tomek, I.: An experiment with the edited nearest-neighbor rule. *IEEE Trans. Syst. Man Cybern.* 6(6), 448–452 (1976)
22. Kopolowitz, J., Brown, T.A.: On the relation of performance to editing in nearest neighbor rules. *Pattern Recognit.* 13(3), 251–255 (1981)
23. Jiang, Y., Zhou, Z.: Editing training data for knn classifiers with neural network ensemble. In: Yin, F.-L., Wang, J., Guo, C. (eds.) *ISNN 2004. LNCS*, vol. 3173, pp. 356–361. Springer, Heidelberg (2004)
24. Sánchez, J.S., Barandela, R., Marqués, A.I., Alejo, R., Badenas, J.: Analysis of new techniques to obtain quality training sets. *Pattern Recognit. Lett.* 24(7) (2003)
25. Delany, S.J., Cunningham, P.: An analysis of case-base editing in a spam filtering system. In: Funk, P., González Calero, P. (eds.) *ECCBR 2004. LNCS (LNAI)*, vol. 3155, pp. 128–141. Springer, Heidelberg (2004)
26. Pan, R., Yang, Q., Pan, S.J.: Mining competent case bases for case-based reasoning. *Artif. Intell.* 171(16–17), 1039–1068 (2007)
27. Angiulli, F.: Fast nearest neighbor condensation for large data sets classification. *IEEE Trans. Knowl. Data Eng.* 19(11), 1450–1464 (2007)
28. Bottou, L., Vapnik, V.: Local learning algorithms. *Neural Comput.* 4(6) (1992)
29. Vapnik, V.N., Bottou, L.: Local algorithms for pattern recognition and dependencies estimation. *Neural Comput.* 5(6), 893–909 (1993)
30. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: *Adv. in Large Margin Classifiers*, pp. 61–74 (1999)
31. Lin, H.T., Lin, C.J., Weng, R.: A note on Platt’s probabilistic outputs for support vector machines. *Mach. Learn.* 68(3), 267–276 (2007)
32. Beygelzimer, A., Kakade, S., Langford, J.: Cover Trees for Nearest Neighbor. In: *ICML 2006*, pp. 97–104. ACM Press, New York (2006)
33. Krauthgamer, R., Lee, J.: Navigating nets: simple algorithms for proximity search. In: *SODA 2004*, Society for Industrial and Applied Mathematics, pp. 798–807 (2004)
34. Chang, C.C., Lin, C.J.: *LIBSVM: a library for support vector machines* (2001)
35. Asuncion, A., Newman, D.J.: *Uci machine learning repository* (2007)
36. Segata, N., Blanzieri, E.: Fast local support vector machines for large datasets. In: *Proc. of MLDM (2009)* (accepted for publication)
37. Uzilov, A., Keegan, J., Mathews, D.: Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC Bioinf.* 7(1), 173 (2006)
38. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* 7, 1–30 (2006)

# Conceptual Neighborhoods for Retrieval in Case-Based Reasoning

Ben G. Weber and Michael Mateas

University of California, Santa Cruz  
Santa Cruz, CA 95064, USA  
{bweber,michaelm}@soe.ucsc.edu

**Abstract.** We present a case-based reasoning technique based on conceptual neighborhoods of cases. The system applies domain knowledge to the case retrieval process in the form of recall and generalize methods. Recall methods utilize domain specific preconditions and perform exact matching, while generalize methods apply transformations that generalize features in queries. The system uses a similarity function based on edit distances, where an edit distance considers only a subset of the features. This retrieval strategy enables the system to locate conceptually similar cases within the feature space. We demonstrate the performance of this approach by applying it to build-order selection in a real-time strategy game. Our results show that the system outperforms nearest neighbor retrieval when enforcing imperfect information in a real-time strategy game.

## 1 Introduction

One of the main challenges in case-based reasoning is constructing effective case representations and retrieval techniques. A common retrieval strategy involves using feature vectors for case descriptions and applying similarity functions for retrieval. The advantage of this approach is that techniques from the machine learning literature can be applied to case-based reasoning. However, the drawbacks of this approach are that it requires a comprehensive example set in order to achieve good results and is sensitive to noise [1].

Case retrieval can be improved by applying domain knowledge to case representation and retrieval. This can be achieved through the use of structural representations or deep features. Using symbolic representations enables case-based reasoning to integrate with other symbolic systems, such as planning. The main advantage of this approach is that a richer case representation enables domain specific retrieval and adaptation methods. The drawbacks of this approach are that it is computationally expensive and often requires annotated examples. Additionally, it can be difficult to encode spatial and temporal domain knowledge structurally.

We present a case-based reasoning technique that maps examples to conceptual neighborhoods of cases. The system uses a basic feature vector representation, but applies domain specific retrieval and adaptation methods. This enables

domain knowledge to be described symbolically, rather than solely through feature vector weighting. This approach is suitable for knowledge rich domains that are difficult to represent structurally.

We apply conceptual neighborhoods to build-order selection in a real-time strategy (RTS) game. Build-order selection is a knowledge rich aspect of RTS games that incorporates temporal reasoning. Cases for build order can be extracted directly from game traces. The case-based reasoning system is integrated with the reactive planning agent of McCoy and Mateas [2]. Retrieval using conceptual neighborhoods is compared with variations of nearest neighbor while enforcing imperfect information in a RTS game.

The remainder of this paper is structured as follows: in the next section we discuss retrieval strategies for case-based reasoning. Section 3 introduces our approach to case retrieval using conceptual neighborhoods. We then apply conceptual neighborhoods to an RTS game in Section 4. Section 5 provides an overview of the system implementation and Section 6 reports our results. We compare our approach to previous work on case-based reasoning in RTS games in Section 7. Finally, we provide conclusions and future work in Section 8.

## 2 Retrieval in Case-Based Reasoning

Case retrieval selects cases based on a similarity metric. Nearest neighbor retrieval evaluates similarity by projecting cases in feature space and computing a distance between points. Structural approaches evaluate similarity by computing the number of transformations needed to translate cases to match a given query.

### 2.1 Nearest Neighbor Retrieval

Nearest neighbor is a form of instance-based learning [3] that has been applied to classification problems and case retrieval in case-based reasoning systems. Nearest neighbor utilizes cases with a feature vector representation. Given a query, nearest neighbor retrieves the nearest case within the feature space. Similarity between cases is computed using a distance function.

The general form for computing the distance between a query,  $q$ , and a case,  $c$ , is defined by the  $L_p$  norm:

$$d(q, c) = \left( \sum_{j=1}^n |q_j - c_j|^p \right)^{1/p}$$

where  $q_j$  and  $c_j$  are features and  $n$  is the number features in the case description. This family of distance functions is also known as the Minkowski distance [4]. Common  $L_p$  norms are  $L_1$ , Manhattan distance, and  $L_2$ , Euclidean distance. Domain specific similarity functions can also be used for nearest neighbor retrieval. For example, edit distance, which computes the number of modifications needed to translate a case into the query, can be augmented with specific knowledge about data to produce knowledge-based similarity measures [5].



Nearest neighbor is sensitive to irrelevant and noisy features [1]. Variations of nearest neighbor have been developed to reduce these issues. Wettschereck and Aha [1] introduce a framework for automating the process of weighting features, which assigns low weights to irrelevant features. Bergmann and Vollrath show that a case can cover a region rather than a point in feature space [6]. They explore the use of generalized cases and present similarity functions for this representation. Another variation of nearest neighbor is neighborhood counting [7]. Wang defines neighborhoods as regions in feature space. To measure the distance between two data points, the similarity function computes the number of neighborhoods that cover a case and the query.

## 2.2 Structural Retrieval

Structural cases provide richer representations for case-based reasoning. Cases are commonly encoded as graphs, where the concepts of the problem domain are represented as nodes and relations between concepts are represented as edges. Edges can represent spatial, temporal, or causal relationships between nodes. Bunke and Messmer [8] introduce a similarity measure based on a weighted graph edit distance.

Structural representations enable case-based reasoning systems to perform problem solving, rather than just classification. For example, MINSTREL is an author-modeling story generator [9] that uses symbolic case-based reasoning. The system has a knowledge base of King Arthur stories and general knowledge about characters in this domain. MINSTREL's case representation enables the system to invent new stories that satisfy character and story goals.

Problem solving in MINSTREL uses knowledge representations known as Transform-Recall-Adapt Methods (TRAM). TRAMs are bundled knowledge representations that know how to transform a problem into a related problem, recall a solution to the new problem and adapt the solution back to the original problem. An example TRAM is Cross-Domain-Solution, which ontologically maps a problem into a new domain, solves the problem in that domain and adapts the solution by reversing the mapping. MINSTREL also uses TRAMs recursively. When performing recursive problem solving, MINSTREL transforms the original problem with multiple TRAMs, recalls a solution to the new problem and applies the adaptation step of the TRAMs to the recalled solution.

## 3 Conceptual Neighborhoods

We present a case-based reasoning system based on conceptual neighborhoods. The system is a hybrid approach between nearest neighbor retrieval and symbolic case-based reasoning. It shares with nearest neighbor methods the use of feature vector representations, while sharing with symbolic case-based reasoning the use of domain specific transform and recall rules.

Conceptual neighborhoods provide a way to organize cases using deep features. Representations based on conceptual neighborhoods have been shown to

allow for reasoning on imprecise knowledge [10]. Conceptual neighborhoods have been applied to case-based reasoning in the legal domain. Hypo [11] uses claim lattices to represent conceptual neighborhoods of cases and exploit connections among cases relevant to the current problem. Conceptual neighborhoods can also be applied to case-based reasoning systems that use a feature vector representation by mapping surface features to deep features. Generalizing a deep feature in this representation enables exploration of a neighborhood of cases.

Our approach maps features to concepts and projects cases in concept space. Concepts can be composed of several features, resulting in a dimensionality reduction. This process is similar to systems that map surface features to deep or knowledge-intensive features [12]. The goals of this mapping are to reduce the effects of noise and enable generalization for case retrieval.

### 3.1 Case Retrieval

Case-based reasoning with conceptual neighborhoods resembles problem solving using TRAMs [9]. However, our approach differs from MINSTREL in that our system does not bound transformations to specific recall methods. An overview of the process is shown in Figure 1. First, the transform step selects 0 to  $n$  generalize methods and applies them to the query, where  $n$  is the maximum number of generalizations allowed. Next, the recall step performs matching using a set of recall methods. Then the system evaluates the recalled cases by computing a distance metric based on the applied generalize methods. Finally, a case is selected from the set of recalled cases.

Recall methods perform exact matching using a subset of the concepts. Concepts that are marked as generalized do not require an exact match, but incur a cost based on a distance metric. The subset of concepts to select is domain specific and is derived from domain knowledge. Matching functions can test for equivalence, greater-than or less-than relations or a domain specific matching function. Recall methods match only on cases with the corresponding class or behavior. Therefore, each recall method matches against a disjoint subset of the case library. Recall methods contain preconditions, which verify that retrieved cases are valid given the query.

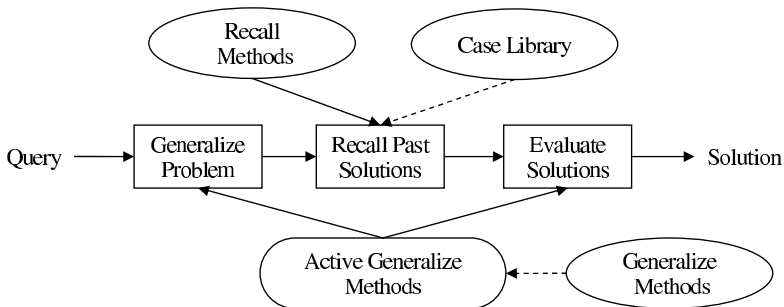


Fig. 1. Retrieval with conceptual neighborhoods

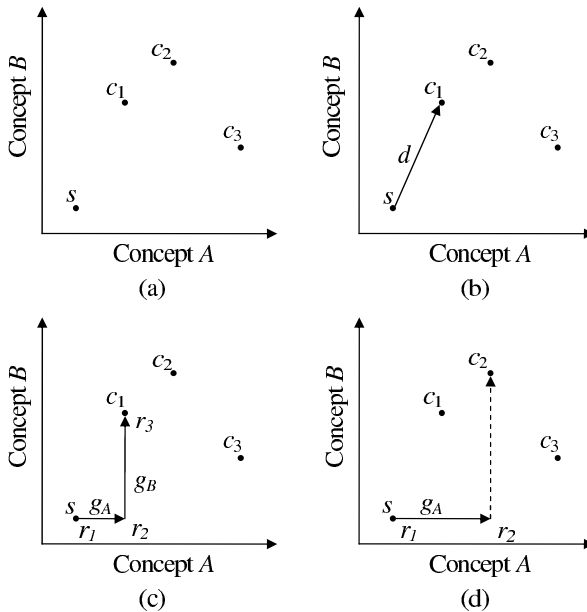
Preconditions for recall methods can specify additional domain knowledge to improve recall performance. The use of recall methods enables the system to evaluate feature subsets based on possible solutions, which differs from previous work [12] that selects feature subsets based on the problem.

Generalize methods transform the query by flagging an individual concept in the query as generalized. There is a generalize method for each concept in the query that can be generalized. Generalize methods enable the system to search the problem space and solution space [9].

The evaluation step computes a distance metric for a case by summing the distance metrics of each generalize method applied to the case. Generalize methods compute an edit distance [8] for the generalized concept between the query and recalled case. The edit distance can be based on individual features mapped to the concept. The distance is zero if the generalized concept is not contained in the subset of concepts used by the recall method that selected the case. The evaluation step then selects a case using a selection strategy, such as highest similarity or weighted random selection.

### 3.2 Retrieval in Concept Space

The conceptual neighborhood approach maps features to concepts and retrieves cases in concept space. An example query is shown in Figure 2. The first graph (a) shows the query,  $s$ , and three cases. The second graph (b) demonstrates



**Fig. 2.** Retrieval in concept space (a) The game state,  $s$ , and three cases (b) Retrieving  $c_1$  using nearest neighbor (c) Retrieving  $c_1$  using conceptual neighborhoods (d) Retrieving  $c_2$  using conceptual neighborhoods

retrieval using nearest neighbor. The distance for retrieving  $c_1$  using nearest neighbor is computed as follows:

$$d = \sqrt{(c_{1A} - s_A)^2 + (c_{1B} - s_B)^2}$$

The remaining graphs demonstrate retrieval using conceptual neighborhoods. The steps to retrieve  $c_1$  in the third graph (c) are the following:

1. Recall fails at  $r_1$ , because  $c_{1A} \neq s_A$
2. Generalize method  $g_A$  generalizes concept  $A$
3. Recall fails at  $r_2$ , because  $c_{1B} \neq s_B$
4. Generalize method  $g_B$  generalizes concept  $B$
5. Recall succeeds at  $r_3$
6.  $d = distance(s_A, c_{1A}) + distance(s_B, c_{1B})$

where  $distance(s_j, c_j)$  is a domain specific edit distance. The steps to retrieve  $c_2$  in the fourth graph (d) are the following:

1. Recall fails at  $r_1$ , because  $c_{2A} \neq s_A$
2. Generalize method  $g_A$  generalizes concept  $A$
3. Recall succeeds at  $r_2$ , because the recall method for  $c_2$  does not consider concept  $B$
4.  $d = distance(s_A, c_{2A})$

Note that different recall methods were used to retrieve  $c_1$  and  $c_2$ . The recall method used to retrieve  $c_1$  matched against both concepts, while the recall method used to retrieve  $c_2$  matched against only concept  $A$ .

### 3.3 Applying Conceptual Neighborhoods

Conceptual neighborhoods can be applied to case-based reasoning systems that use a feature vector representation. The first step is to select a set of concepts and map the original features to concepts. The next step is to create a recall method for each class or behavior in the domain. The third step is to select concept subsets for each recall method. The final step is to select which concepts can be generalized and to determine edit distances for these concepts. This process is demonstrated in the next section.

## 4 Conceptual Neighborhoods in RTS Games

In this section we describe how conceptual neighborhoods can be applied to build order in Wargus<sup>1</sup>, a clone of the game Warcraft II which was developed by Blizzard Entertainment<sup>TM</sup>. The purpose of the case-based reasoner is to select the next unit or building to produce based on the current game state.

RTS games present a variety of research problems, including decision making under uncertainty, opponent modeling and adversarial planning [13]. RTS games

<sup>1</sup> <http://wargus.sourceforge.net>

enforce imperfect information through a “fog of war”, which limits visibility to portions of the map where the player controls units. In order to acquire information about an opponent, it is necessary to actively scout the map to find out which buildings and units the opponent is producing. Scouting is vital in RTS games, because different strategies have different types of counter strategies.

One of the focuses of strategic play in RTS games is build order. A build order defines the sequence in which buildings are constructed, units are produced and technologies are researched. Build order is a knowledge-rich aspect of RTS gameplay and players can improve their skills by studying replays of professional matches and learning which build orders work best against counter strategies on a variety maps.

#### 4.1 Case Representation

We define a case as a behavior and game state pair. Behaviors are discussed in more detail in the following section. Game state includes the following concepts: player technological state (player tech), enemy technological state (enemy tech), number of combat units, number of workers, number of production buildings and map properties. The mapping of features to concepts is shown in Table 1.

#### 4.2 Recall Methods

The build order behaviors in Wargus can be classified by the following actions: train worker unit, train combat unit, build tech building, build production building and research upgrade. The system contains a recall method for each behavior type. The subsets of concepts evaluated by the recall methods are shown in Table 2. The following concepts require an exact match between the query and a case: map properties, player tech, enemy tech and number of production buildings. The number of workers and number of combat units concepts require that the query contains at least as many units as a case.

We selected the concept subsets for each recall method based on analyzing expert replays. Domain knowledge is demonstrated by the research-upgrade recall method, which matches against only the player tech and number of combat unit concepts. If the player possesses several combat units and the tech buildings required to research an upgrade, then researching the upgrade is a preferred action.

**Table 1.** Game state features mapped to concepts

Concept	Features
Player Tech	Lumber Mill, Blacksmith, Stronghold, Mound
Enemy Tech	Enemy Barracks, Lumber Mill, Blacksmith, Stronghold, Mound
Prod. Buildings	Barracks
Workers Units	Peons
Combat Units	Grunts, Axe throwers, Catapults, Ogres
Map Properties	Distance to opponent base, Open path to opponent base

**Table 2.** Concept subsets for recall methods

	Map properties	Player tech	Enemy tech	Worker units	Combat units	Production buildings
Train worker	✓	✓	✓	✓	✓	
Train combat unit	✓	✓	✓	✓		✓
Tech building	✓	✓	✓	✓		
Production building	✓	✓	✓	✓	✓	✓
Research upgrade		✓			✓	

### 4.3 Generalize Methods

The system includes a generalize method for each concept in the case representation, excluding the player tech concept. Generalize methods mark a concept as not requiring an exact match at the cost of an edit distance. The edit distance computes the distance between the query and recalled cases based on the amount of in-game resources required to translate the query to a recalled case for a particular concept, by computing a linear combination of the gold and wood resources.

**Generalize number of workers** marks the number of workers concept as generalized. The edit distance is the cost of a worker unit times the difference in number of worker units between the query and a recalled case.

**Generalize number of production buildings** marks the number of production buildings concept as generalized and computes the edit distance based on the difference in number of production buildings times the cost of a production building.

**Generalize number of combat units** marks the number of combat units concept as generalized and computes the distance based on the difference in combat units between the query and recalled case. Although the number of combat units concept is an aggregation, the distance is computed based on the individual unit types. Therefore, the distance metric distinguishes between expensive and inexpensive units.

**Generalize enemy tech** computes a distance metric based on the difference in enemy tech buildings between the query and recalled case. The distance metric sums the cost of the buildings that are different.

**Generalize map property** causes recall methods to ignore map properties when retrieving cases. The distance metric is a constant cost and is incurred if any of the map properties differ between the query and a recalled case.

### 4.4 Case Selection

A case is selected from the set of retrieved cases using weighted random selection. Weights are computed using an inverse distance relation:

$$weight = \frac{1}{\delta + \sum_{j=1}^n distance_j(q_j, c_j)}$$

where  $q_j$  is a concept of the query,  $c_j$  is a concept of a case,  $distance_j$  is the edit distance for concept  $j$ ,  $n$  is the number of concepts and  $\delta$  is a constant to adjust so the value is finite if the edit distance is zero.

A randomized selection is used to enable constrained exploration of the case library. This leads to small variations in build order. Also, always picking the best case can cause problems in an imperfect information environment, because noise can cause the similarity function to be inaccurate. The system uses an inverse distance relation, but other approaches could be used, such as exponential weighting.

#### 4.5 Retrieval Example

An example query with three cases is shown in Figure 3. The cases correspond to training a peon ( $c_1$ ), building a blacksmith ( $c_2$ ) and building a barracks ( $c_3$ ). In Wargus, the cost of a worker unit is 400 gold and the cost of a first-tier combat unit is 600 gold. Retrieving  $c_1$  consists of the following steps:

1. Recall fails at  $r_1$ , because  $c_{1w} \neq s_w$
2.  $g_w$  generalizes the number of worker units
3. Recall fails at  $r_2$ , because  $c_{1c} \neq s_c$
4.  $g_c$  generalizes the number of combat units
5. Recall succeeds at  $r_3$
6.  $d_{c1} = distance(s_w, c_{1w}) + distance(s_c, c_{1c}) = 1 * 400 + 2 * 600 = 1600$

Retrieving  $c_2$  consists of the following steps:

1. Recall fails at  $r_1$ , because  $c_{2w} \neq s_w$
2.  $g_w$  generalizes the number of worker units
3. Recall succeeds at  $r_2$
4.  $d_{c2} = distance(s_w, c_{2w}) = 2 * 400 = 800$

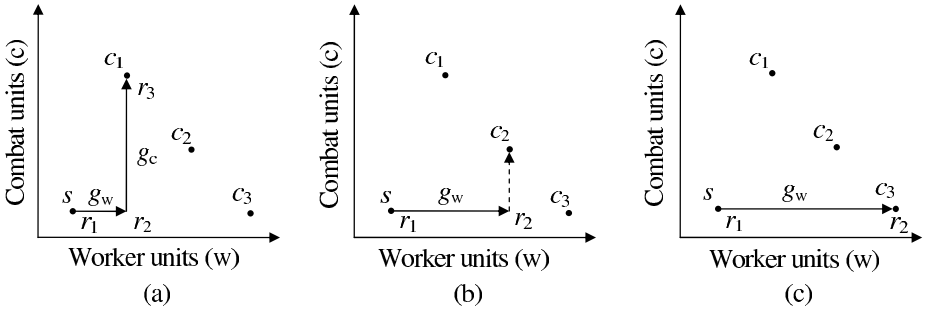
Retrieving  $c_3$  consists of the following steps:

1. Recall fails at  $r_1$ , because  $c_{3w} \neq s_w$
2.  $g_w$  generalizes the number of worker units
3. Recall succeeds at  $r_2$
4.  $d_{c3} = distance(s_w, c_{3w}) = 3 * 400 = 1200$

Weights are computed based on these distances. Setting  $\delta = 400$  results in the following weights:  $w_{c1} = 0.0005$ ,  $w_{c2} = 0.00083$  and  $w_{c3} = 0.00063$ . These weights are used to perform a weighted random selection.

## 5 Implementation

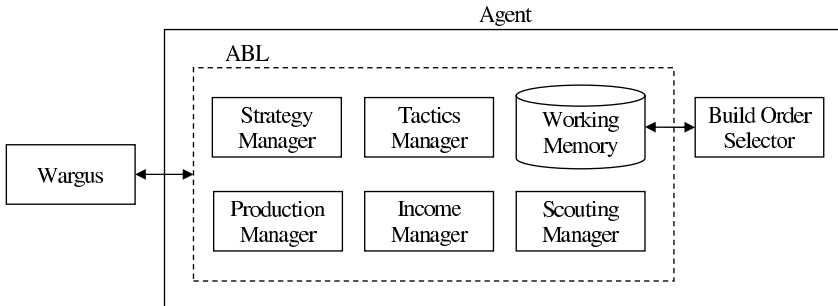
Our system uses the integrated agent framework of McCoy and Mateas [2]. The case-based reasoning system communicates with the framework using the blackboard pattern. McCoy and Mateas' agent was modified to produce buildings and units based on events posted to the blackboard. Reconnaissance capabilities were also added to the agent. Six case retrieval strategies were implemented to evaluate the performance of conceptual neighborhoods.



**Fig. 3.** An example query. The cases correspond to training a peon (a), building a blacksmith (b) and building a barracks (c).

### 5.1 Architecture

The game-playing agent consists of an ABL agent connected to the Wargus RTS engine. A behavior language (ABL) is a reactive planning language [14] and communicates with Wargus using JNI. Different competencies in the agent communicate with each other through ABL’s working memory. ABL’s working memory serves as a blackboard and enables communication through the blackboard pattern. An overview of the agent architecture is shown in Figure 4.



**Fig. 4.** Agent architecture

The agent is composed of distinct managers, each of which is responsible for performing one or more subtasks. The strategy manager is responsible for high-level strategic decisions and focuses on build order. The production manager is responsible for producing units and buildings, based on messages generated by the strategy manager. The tactics manager decides when and where to engage the opponent. The scouting manager is responsible for assigning worker units to scout the opponent base.



## 5.2 Build Order Selectors

Six retrieval strategies were implemented for the build order selector. The goal of implementing several retrieval strategies was to determine if retrieval with conceptual neighborhoods outperforms nearest neighbor and to evaluate if both generalize and recall methods are necessary for the conceptual neighborhood approach to be effective.

**Random build order selector (Rand)** picks cases randomly from the set of valid cases.

**Nearest neighbor selector (NNS)** performs case retrieval using Manhattan distance. The case description contains a feature for each unit type, for both players.

**Case feature selector (CFS)** performs case retrieval using the concepts and edit distances discussed in the previous section, but does not use generalize or recall methods.

**Generalize methods only selector (GMS)** performs exact matching using generalize methods. This approach does not use concept subsets for recall.

**Recall methods only selector (RMS)** performs partial matching using recall methods and concept subsets.

**Conceptual neighborhood selector (CNS)** uses both generalize and recall methods to perform case retrieval.

## 5.3 Case Generation

A case library was generated by running several different scripted builds against each other on several different maps. The scripts were selected from eight hand-coded build orders with specific timing attacks. The validation scripts, discussed in the results section, were not used for case generation. The map pool consisted of maps with varying distances between bases (close, medium, far) and open and closed paths between bases. Four scripts were selected for each map and tested against the other scripts, for a total of six games per map. Cases were added to the library only for the winning script. The case library contains 36 games traces and over 1500 cases. The case library is much larger than previous work utilizing game traces [15,16].

## 6 Results

The agent was evaluated against the built-in AI of Wargus, two well-established scripts and a new script. Four different maps were used, where the first three maps contain a direct land route to the opponent's base of varying distance (close, medium, far) and the last map (NWTR) is a variation of the map "Nowhere to run, nowhere to hide". The agent was tested in perfect and imperfect information environments. In games with perfect information, the game state is fully observable. In games with imperfect information, the "fog of war" is enforced, limiting the visibility of the agent to areas where units are controlled.

**Table 3.** Win rates for perfect and imperfect information environments over 32 trials

	Perfect Information	Imperfect Information
Rand	31%	19%
NNS	69%	50%
CFS	56%	44%
GMS	44%	41%
RMS	50%	47%
CNS	75%	66%

**Table 4.** Win rates versus scripted builds over 8 trials

	Land Attack	Soldier's Rush	Knight's Rush	Fast Ogre	Overall
Rand	62%	12%	0%	0%	19%
NNS	62%	62%	38%	38%	50%
CFS	100%	50%	12%	12%	44%
GMS	75%	50%	25%	12%	41%
RMS	88%	62%	25%	12%	47%
CNS	100%	75%	50%	38%	66%

Games were run with perfect and imperfect information and results are shown in Table 3. The conceptual neighborhood selector won 66% of games in an imperfect information environment. Also, the success rate of the conceptual neighborhood selector decreased by only 9% when enforcing imperfect information, while the success rate of the nearest neighbor selector decreased by 19%.

Win rates against the scripted builds with imperfect information enforced are shown in Table 4. The conceptual neighborhood selector outperformed all of the other selectors. All of the retrieval strategies outperformed random selection, but the case feature, generalize methods only, and recall methods only selectors performed worse than nearest neighbor retrieval. The conceptual neighborhood selector achieved a success rate of at least 50% on every map (see Table 5). These results indicate that the conceptual neighborhood approach was better at adapting to new game situations.

**Table 5.** Win rates on the map pool over 8 trials

	Open Close	Open Medium	Open Far	NWTR
Rand	25%	25%	25%	0%
NNS	62%	12%	88%	25%
CFS	38%	50%	50%	38%
GMS	50%	12%	62%	38%
RMS	50%	50%	62%	25%
CNS	75%	62%	75%	50%

## 7 Related Work

Case-based reasoning has been applied to several aspects of RTS play, including strategic [17] and tactical [18] levels of gameplay. There are two approaches that have been applied to case-based reasoning in RTS games: bootstrap learning systems that rely on exploration of the state space and systems that utilize game traces to automatically acquire knowledge about RTS gameplay.

Aha et al. [17] use case-based reasoning to defeat strategies randomly selected from a pool of fixed strategies. The system uses the building-specific state lattice developed by Ponsen et al. [19], which abstracts Wargus game states into a state lattice and specifies a set of counter strategies for each state. This knowledge is used to explore the state space and build a case library of counter strategies. Our system differs in that a state lattice is not used to constrain the set of possible strategies.

Molineaux et al. [18] apply case-based reasoning to tactical situations in RTS games. They claim that tactical gameplay in RTS games is knowledge poor and therefore a state-space taxonomy is insufficient to encompass all relevant tactical decisions. They combine case-based reasoning and reinforcement learning in order to explore the state space. Our system varies from this approach, because our system makes use of domain knowledge.

Game traces have been used by case-based reasoning systems to play full RTS games. Ontañón et al. [15] present a case-based planning approach that uses game traces to interleave planning and execution, and play at the same action granularity as a human player. The system uses expert-annotated game traces to automatically acquire domain knowledge from expert players. Cases are extracted from traces and specify primitive actions or additional subgoals for the current behavior to pursue. Our approach differs in that we define a clear division between planning and case-based reasoning and case-based reasoning is applied only to build order.

Mishra et al. [12] extend the case-based planner by introducing situation assessment for improved case retrieval. They noticed that the performance of previous systems [15,16] suffers when the case library stores numerous plans representing several strategies played over maps of different sizes. Mishra et al. introduce the concept of a situation and use situation assessment to aid in case retrieval. A situation is defined by a high-level representation of the game state including map properties and current goals of the player. Situations are then used to select relevant features for case retrieval. The main difference between this approach and our system is that Mishra et al. select feature subsets based on the current game state, while the conceptual neighborhood approach selects feature subsets based on the type of case being recalled. Additionally, previous work [15,16,12] has investigated smaller numbers of game traces, which also required annotation.

Our results are compared to reported success rates from the literature in Table 6. All prior work, to our knowledge, has used perfect information. We report results for the performance of the system in perfect and imperfect information environments, which achieved the same win rates against the standard

**Table 6.** Reported win rates versus the conceptual neighborhood selector with perfect information (PI) and imperfect information (II)

	Ponsen et al.	Ontañón et al.	McCoy& Mateas	CNS PI	CNS II
Land Attack	76%	89%	—	100%	100%
Soldier's Rush	29%	—	80%	75%	75%
Knight's Rush	13%	—	53%	50%	50%

scripts. Aha et al. [17] report an average success rate of over 80%, but do not specify win rates against the soldier's and knight's rushes.

## 8 Conclusion

In this paper we have demonstrated how conceptual neighborhoods can be applied to retrieval in case-based reasoning. Our contributions include the application of conceptual neighborhoods to retrieval, which enables additional domain knowledge to be applied to case retrieval, and evaluation of conceptual neighborhoods versus other retrieval strategies. We validated our approach by applying conceptual neighborhoods to build order in a RTS game. The results indicate that retrieval using conceptual neighborhoods outperforms nearest neighbor when enforcing imperfect information.

Our results show two interesting properties. First, the conceptual neighborhood approach achieved similar success rates to nearest neighbor retrieval in a perfect information environment, while outperforming nearest neighbor retrieval when imperfect information is enforced. This leads us to conclude that the conceptual neighborhood approach is better at adapting to new game situations. Second, the approaches that did not use both generalize and recall methods performed worse than nearest neighbor retrieval. If the generalize methods only selector had outperformed nearest neighbor, then the success of the system could be attributed to the use of domain specific distance metrics. However, our results show that the use of concept subsets was necessary to improve retrieval. This indicates that the interaction between generalize and recall methods is necessary to capture domain knowledge for retrieval with conceptual neighborhoods.

Future work will explore different types of transformations and the application of conceptual neighborhoods to additional aspects of case-based reasoning, including case adaptation and on-line learning.

## References

1. Wettschereck, D., Aha, D.: Weighting features. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 347–358. Springer, Heidelberg (1995)
2. McCoy, J., Mateas, M.: An Integrated Agent for Playing Real-Time Strategy Games. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Chicago, Illinois, pp. 1313–1318. AAAI Press, Menlo Park (2008)

3. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. *Machine Learning* 6(1), 37–66 (1991)
4. Bagherjeiran, A., Eick, C.F.: Distance function learning for supervised similarity assessment. In: *Case-Based Reasoning on Images and Signals*, pp. 91–126. Springer, Heidelberg (2008)
5. Cunningham, P.: A taxonomy of similarity mechanisms for case-based reasoning. *IEEE Transactions on Knowledge and Data Engineering* (forthcoming)
6. Bergmann, R., Vollrath, I.: Generalized Cases: Representation and Steps Towards Efficient Similarity Assessment. In: Burgard, W., Christaller, T., Cremers, A.B. (eds.) *KI 1999. LNCS*, vol. 1701, pp. 195–206. Springer, Heidelberg (1999)
7. Wang, H.: Nearest Neighbors by Neighborhood Counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(6), 942–953 (2006)
8. Bunke, H., Messmer, B.: Similarity Measures for Structured Representations. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) *EWCBR 1993. LNCS*, vol. 837, pp. 106–118. Springer, Heidelberg (1994)
9. Turner, S.R.: *The Creative Process: A Computer Model of Storytelling and Creativity*. Lawrence Erlbaum Associates, Mahwah (1994)
10. Freksa, C.: Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence* 54(1), 199–227 (1992)
11. Ashley, K., Rissland, E.: A case-based approach to modeling legal expertise. *IEEE Expert: Intelligent Systems and Their Applications* 3(3), 70–77 (1988)
12. Mishra, K., Ontañón, S., Ram, A.: Situation assessment for plan retrieval in real-time strategy games. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) *ECCBR 2008. LNCS*, vol. 5239, pp. 355–369. Springer, Heidelberg (2008)
13. Buro, M.: Real-Time Strategy Games: A New AI Research Challenge. In: *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico*, pp. 1534–1535. Morgan Kaufmann, San Francisco (2003)
14. Mateas, M., Stern, A.: A Behavior Language for Story-Based Believable Agents. *IEEE Intelligent Systems* 17(4), 39–47 (2002)
15. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-Based Planning and Execution for Real-Time Strategy Games. In: Weber, R.O., Richter, M.M. (eds.) *ICCBR 2007. LNCS*, vol. 4626, pp. 164–178. Springer, Heidelberg (2007)
16. Sugandh, N., Ontañón, S., Ram, A.: On-Line Case-Based Plan Adaptation for Real-Time Strategy Games. In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Chicago, Illinois*, pp. 702–707. AAAI Press, Menlo Park (2008)
17. Aha, D., Molineaux, M., Ponsen, M.: Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In: Muñoz-Ávila, H., Ricci, F. (eds.) *ICCBR 2005. LNCS*, vol. 3620, pp. 5–20. Springer, Heidelberg (2005)
18. Molineaux, M., Aha, D.W., Moore, P.: Learning Continuous Action Models in a Real-Time Strategy Environment. In: *Proceedings of the Twenty-First Florida Artificial Intelligence Research Conference, Coconut Grove, Florida*, pp. 257–262. AAAI Press, Menlo Park (2008)
19. Ponsen, M.J.V., Muñoz-Avila, H., Spronck, P., Aha, D.W.: Automatically Acquiring Domain Knowledge For Adaptive Game AI Using Evolutionary Learning. In: *Proceedings of the Twentieth National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania*, pp. 1535–1540. AAAI Press, Menlo Park (2005)

# CBR Supports Decision Analysis with Uncertainty

Ning Xiong and Peter Funk

School of Innovation, Design and Engineering  
Mälardalen University  
SE-72123 Västerås, Sweden  
{Ning.Xiong, Peter.Funk}@mdh.se

**Abstract.** This paper proposes a novel approach to case-based decision analysis supported by case-based reasoning (CBR). The strength of CBR is utilized for building a situation dependent decision model without complete domain knowledge. This is achieved by deriving states probabilities and general utility estimates from the case library and the subset of cases retrieved in a situation described in query. In particular, the derivation of state probabilities is realized through an information fusion process which comprises evidence (case) combination using the Dempster-Shafer theory and Bayesian probabilistic reasoning. Subsequently decision theory is applied to the decision model learnt from previous cases to identify the most promising, secured, and rational choices. In such a way we take advantage of both the strength of CBR to learn without domain knowledge and the ability of decision theory to analyze under uncertainty. We have also studied the issue of imprecise representations of utility in individual cases and explained how fuzzy decision analysis can be conducted when case specific utilities are assigned with fuzzy data.

**Keywords:** Case-based decision analysis, case-based reasoning, decision model, similarity, basic probability assignment, information fusion.

## 1 Introduction

Decision making is prevalent in solving many engineering, health care and management problems. It has also gained increasing importance for intelligent agent systems [1] to interact with the environment autonomously. The main challenges in most practical decision problems are how to cope with uncertain characteristics in the environment and how to make choices in the presence of these uncertain features. Decision theory [2, 3] has offered useful tools in analyzing uncertain situations to identify the “best” course of actions from a reasonable perspective. However, practical applications of decision theory entail formulating a real world problem into a perfect decision model, which may be hard to achieve in many circumstances due to complexity, poor domain knowledge, as well as incomplete information.

A more pragmatic method to make decisions is to visit previous similar situations as reference. It was argued in [4] that decision making under uncertainty is at least partly case-based. With a case-based method we don't require fully understood

domain knowledge for building a precise decision model. The research into this realm is strongly supported by the methodology of case-based reasoning (CBR) [5]. Recently CBR has been widely employed as decision support for explanation [6, 7], label ranking [8], as well as recommendation and advice giving [9-13] in numerous practical applications.

This paper proposes a novel approach to support decision analysis using CBR methods. The power of CBR is utilized for creating a situation dependent decision model from past similar experiences. Further, decision theory is applied to the decision model learnt from past experiences to find out optimal, rational, and low risk solutions. With such integration we create a unified framework in which CBR and decision theory can complement each other. CBR helps decision analysis dealing with complicated problems with poor domain knowledge and incomplete information, while decision theory helps CBR handling uncertain information and features in the problem domain.

The kernel of the proposed work is the case-based learning of a decision model. This is a bit different from the common practice in many CBR systems where finding solutions to the query case appears the main goal of the CBR task. What we seek here is to derive, from previous experiences, a probabilistic characterization of the current situation in terms of likelihoods, risks and probable consequences. We hope this would offer a useful means to tackle the inherent nature of uncertainty in a CBR process, in particular when similar situations don't have similar solutions.

The paper is organized as follows. Section 2 outlines the proposed approach for case-based decision analysis at a general level. We explain derivation of state probabilities for a query situation in section 3, which is followed by estimation of general utilities of actions under states in section 4. Then, in section 5, we discuss decision analysis based on a decision model learnt from cases. Section 6 presents some related work. Finally this paper is concluded in section 7.

## 2 Case-Based Decision Analysis: The Proposed Approach

This section outlines the proposed approach for case-based decision analysis. We start with basics about the decision tree as a decision model. We shall then present the general idea of creating a decision tree from cases to support decision analysis.

### 2.1 Decision Model for Decision Analysis

The decision problem for an agent can be abstracted as follows. Given an environment with possible states  $s_1, s_2, \dots, s_n$ , the agent has to make a choice from a set of alternative actions  $\{a_1, a_2, \dots, a_m\}$ . The outcome or consequence of an action is dependent on the real state of the environment. A general utility function has been defined for all possible outcomes regarding actions and states. By  $u_{ij}$  we denote the general utility of performing action  $a_i$  when state  $s_j$  is true, i.e.,  $u_{ij} = U(a_i | s_j)$ . But the agent has no exact knowledge about the state of the environment, only a probability distribution of the states is available for decision analysis.

This (decision) problem can also be modelled as a decision tree as shown in Fig. 1, where  $p_i$  refers to the probability of state  $s_i$  ( $i=1\dots n$ ). The availability of such a model is prerequisite to apply well founded decision analysis methods such as Bayesian decision theory [2] and the principle of general risk constraints [14] for making profitable, secured, and rational choices

However, constructing a perfect decision tree to abstract an underlying situation is not trivial. It requires thorough understanding of the circumstance and detailed domain knowledge for elicitation of all relevant information. In many cases it is hard to define accurate values for probabilities concerning states of the environment and general utilities regarding actions and states in a decision tree. First of all, estimates for probabilities of states are very likely to be subjective or imprecise. It was observed in [15] that most people usually can not distinguish between probabilities roughly ranging from 0.3 to 0.7. Moreover, general utilities regarding actions and states correspond to a sort of generalized information which is hard to explicate without deep domain knowledge. Instead of giving utility in a general sense, users in real life would feel more natural and confident to specify individual utility scores associated with specific cases by evaluation of concrete results therein. Later we will show in the paper how both the state probabilities and the (general) utilities in the decision tree can be estimated from previous cases for a new situation by using a case-based approach.

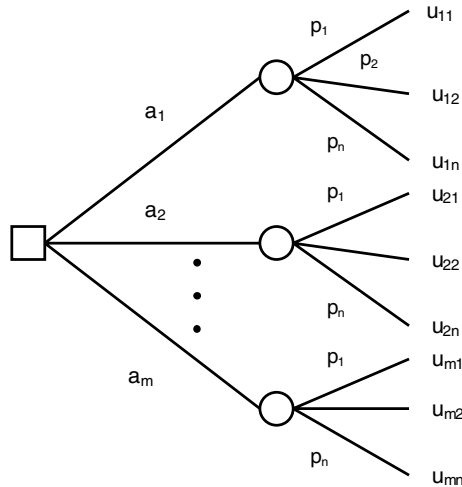


Fig. 1. A decision problem modelled as a decision tree

### 2.2 Case-Based Learning of Decision Trees

We consider decision trees as vehicles for carrying knowledge and information about candidate actions and their probable consequences. The content of the vehicle is situation dependent. In different situations we may have different alternatives, varying probabilities and different consequences. Here we propose a case-based approach to



creating situation dependent decision trees. The basic idea is to derive the right content of the decision model by resorting to previous similar cases with respect to a given new situation. This approach is different from conventional ways CBR works to recommend final solutions based on a subset of retrieved cases. Contrarily, in this paper, we apply CBR in an intermediate stage for creation of a qualified decision model, which can then be utilized by various decision analysis methods to find out rational, justified choices.

A procedure for case-based learning of decision trees is shown in Fig. 2. It starts with similarity matching between a new situation and previous cases in the case library. Every case in the case library receives a similarity score according to a predefined similarity metric. We will not detail the issue of similarity measures due to the scope of this paper, but interested readers can refer to the references [16-19] for recent advancements of similarity modelling in CBR research. After similarity matching, a subset of cases that get the highest similarity scores or pass a specified similarity threshold are selected and retrieved. In the next step, we perform probability and utility derivation based on the subset of retrieved cases and the case library. The purpose is to exploit the information residing in the cases to acquire probabilities of environment states in the current situation as well as (general) utility estimates of alternative actions given different states. Finally, the derived probability and utility values are entered into the decision tree for decision analysis.

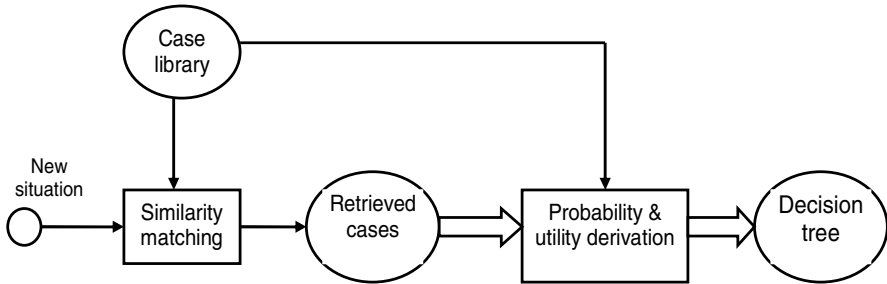


Fig. 2. Case-based learning of decision trees

As basic notation, we assume that a case  $C_j$  in the case library is indexed by a 4-tuple  $C_j=(B_j, E_j, A_j, U_j)$ , where

- $B_j$  is the description of the situation associated with the case. It can, for instance, consist of a set of observed or user-acquired attribute values.
- $E_j=(P_j(s_1), P_j(s_2), \dots, P_j(s_n))$  represents the known probability distribution for states  $s_1, s_2, \dots, s_n$  in the situation associated with the case. States are usually not observable but reflect internal properties of the environment. Sometimes the probability of a state in a case is also notated as  $P_j(s_i) = P(s_i | C_j)$ .
- $A_j$  denotes an action that was performed in the situation associated with the case.
- $U_j$  is an individual utility score evaluating the outcome of performing action  $A_j$  in the situation associated with the case. Hence it is also notated as  $U(A_j|C_j)$  later in the paper.

### 3 Deriving State Probabilities from Previous Cases

The procedure for deriving probabilities of states based upon available cases is depicted in Fig. 3. Given a new target situation  $Q$ , we look for its similar cases in the case library and a subset of cases is retrieved according to the rule of *KNN* ( $k$  nearest neighborhoods) or a specified similarity threshold. The retrieved cases are then delivered along with their similarity degrees to the block “information fusion” for assessing the probabilities of states in the new situation  $Q$ . The information fusion block is further divided into two successive steps, as will be described in subsections 3.1 and 3.2 respectively. The first step concerns evidence combination using the Dempster-Shafer theory (simply D-S theory) [20-21] to yield initial beliefs in states. The D-S theory enables distinguishing different cases in the information fusion process according to their similarity degrees. The second step aims to refine these initial beliefs into final probability evaluations via probabilistic reasoning.

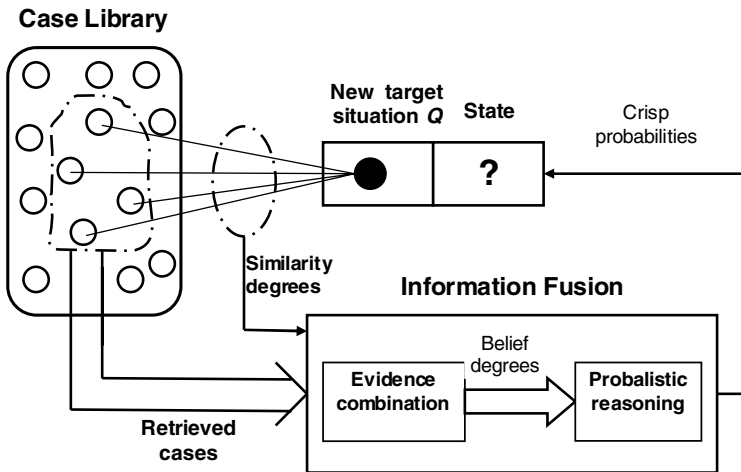


Fig. 3. Derivation of state probabilities based on cases

#### 3.1 Reasoning Degrees of Belief Using the D-S Theory

We consider every retrieved case as a source of information. The evidence combination rule of the Dempster-Shafer theory is employed to aggregate information from relevant cases for assessing the degrees of beliefs in possible states in the query situation.

##### 3.1.1 Evidence Combination Rule of the D-S Theory

The D-S theory is a powerful tool tackling uncertainty. But we do not intend to have an extensive discussion of it in this paper. We shall only introduce some basic concepts of this theory that are relevant for our task of belief aggregation from multiple cases.

In the D-S theory, a sample space of the problem domain is called a “frame of discernment”, notated as  $X$ . It is assumed that one’s total belief due to a piece of evidence can be partitioned into various probability masses, each assigned to a subset of  $X$ . These probability masses are specified by basic probability assignment (BPA), which is a function  $m$  performing mapping from the power set of  $X$  to the interval  $[0, 1]$  satisfying:

$$m(\emptyset) = 0 \tag{1}$$

$$\sum_{F \subseteq X} m(F) = 1 \tag{2}$$

In particular the subsets  $F$  of  $X$  such that  $m(F) > 0$  are called the focal elements of the D-S belief structure.

Owing to imprecision of information, we can not figure out exact probability values for arbitrary subsets of  $X$  from a BPA function. The following two measures are therefore introduced to impose bounds on the probability of a hypothesis.

Let hypothesis  $F$  be a subset of  $X$ , the belief of  $F$ , denoted  $Bel(F)$ , is defined as

$$Bel(F) = \sum_{G \subseteq F} m(G) \tag{3}$$

The plausibility of  $F$ , denoted  $Pl(F)$ , is defined as

$$Pl(F) = \sum_{G \cap F \neq \emptyset} m(G) \tag{4}$$

It was shown in [22] that, for any subset  $F$  of  $X$ , we have the inequality below

$$Bel(F) \leq P(F) \leq Pl(F) \tag{5}$$

This reads that the belief and plausibility measures provide lower and upper bounds on the probability of a hypothesis.

Suppose there are two bodies of evidences over the same frame of discernment, but induced from independent information sources. The BPA functions associated with the two bodies of evidences are  $m_1$  and  $m_2$  respectively. The task now is to combine the evidence related functions  $m_1$  and  $m_2$  into an aggregated (basic) probability assignment function  $m_{12} = m_1 \oplus m_2$ . According to the evidence combination rule of the D-S theory, the basic probability mass for a hypothesis  $F$  ( $F \subseteq X$ ), incorporating both pieces of evidences, is calculated as follows:

$$m_{12}(\emptyset) = 0, \quad m_{12}(F) = K \left( \sum_{F_1 \cap F_2 = F} m_1(F_1)m_2(F_2) \right) \tag{6a}$$

$$K = \left( 1 - \sum_{F_1 \cap F_2 = \emptyset} m_1(F_1)m_2(F_2) \right)^{-1} \tag{6b}$$

The above combination rule reads that  $m_{12}(F)$  is calculated from the summation of the products  $m_1(F_1)m_2(F_2)$  where the intersection of  $F_1$  and  $F_2$  equals  $F$ . The quantity  $K$  plays the role of normalization such that the sum of basic probability numbers for

all subsets of  $X$  equals one.  $K$  is computed on all pairs of  $F_1$  and  $F_2$  that have no intersections with each other. Next we shall use this combination rule to estimate the degrees of belief in various states based on the cases retrieved from the case library.

### 3.1.2 Combining Retrieved Cases as Evidences

Suppose  $Nr$  cases are retrieved from the case library after similarity matching. Without loss of generality, we denote the set of retrieved cases by

$$E = \{C_1, C_2, \dots, C_{Nr}\} \tag{7}$$

The similarity degrees of these retrieved cases against the query situation are given by  $Sim = \{\alpha_1, \alpha_2, \dots, \alpha_{Nr}\}$  where  $\alpha_j$  represents the degree of similarity of case  $C_j$ . Our task here is to aggregate the information of the cases in  $E$  to acquire combined degrees of belief in various states in the current situation.

Obviously the frame of discernment,  $X$ , in our problem domain is the set of states in the environment. In order to apply the evidence combination rule stated above, we first have to interpret the probability distributions in individual cases into a form complying with the D-S belief structure. This can be easily done by restricting the focal elements of the belief structure to individual states as singleton subsets of  $X$ . Hence the probability distribution in a case  $C_j$  can be interpreted as a basic probability assignment function written as

$$BP(C_j) = \{(s_i, P_j(s_i)), i = 1 \dots n\} \tag{8}$$

where  $s_i$  denotes a state in the environment and  $P_j(s_i)$  is the probability that state  $s_i$  is true in the situation described by case  $C_j$ .

Now consider the basic probability assignment function that is induced by the evidence of a retrieved case  $C_j$ . Let  $m(i, j)$  be the basic probability value to which the hypothesis that state  $s_i$  is true is supported by case  $C_j$  as evidence. This probability mass should be reduced from function (8) with similarity degree  $\alpha_j$  as discounting factor. Hence we have

$$m(i, j) = \alpha_j \cdot P_j(s_i) \quad i = 1, \dots, n; \quad j = 1, \dots, Nr \tag{9}$$

As the sum of basic probabilities of states is now smaller than one according to (9), we introduce an extra subset  $S$  containing all possible states. The subset  $S$  receives the remaining probability mass unassigned to any individual state. Thus we can write

$$m(S, j) = 1 - \sum_{i=1}^n m(i, j) = 1 - \sum_{i=1}^n \alpha_j P_j(s_i) = 1 - \alpha_j, \quad j = 1, \dots, Nr \tag{10}$$

Having established basic probability assignments induced by retrieved cases, we now attempt to aggregate these assignment functions into an overall assessment using the evidence combination rule. Denote  $E_t$  as the set of the first  $t$  retrieved cases as follows:

$$E_t = \{C_1, C_2, \dots, C_t\} \tag{11}$$

Let  $m(i, E_t)$  be the basic probability mass to which the hypothesis that state  $s_i$  is true is supported by all evidences (retrieved cases) in  $E_t$ . By  $m(S, E_t)$  we denote the remaining probability mass unassigned to individual states after all evidences in  $E_t$

have been combined. The algorithm to fuse case information according to the evidence combination rule can be formulated in a recursive form as follows:

$$m(i, E_{t+1}) = K_{t+1} (m(i, E_t)m(i, t + 1) + m(i, E_t)m(S, t + 1) + m(S, E_t)m(i, t + 1)) \quad (12a)$$

$$i = 1 \cdots n$$

$$m(S, E_{t+1}) = K_{t+1}m(S, E_t)m(S, t + 1) \quad (12b)$$

$$K_{t+1} = \left( 1 - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n m(i, E_t)m(j, t + 1) \right)^{-1} \quad t = 1 \cdots Nr - 1 \quad (12c)$$

where  $K_{t+1}$  is a normalizing factor to make the sum of the basic probability values induced by the evidences in  $E_{t+1}$  equal one. It bears noting that, to start with the above recursive form, we have  $m(i, E_1) = m(i, I)$  and  $m(S, E_1) = m(S, I)$ . The final outcomes of this combination procedure are  $m(i, E_{Nr})$  and  $m(S, E_{Nr})$ , which correspond to the basic probability values after incorporating all retrieved cases as evidences.

In terms of the belief function defined in (3), the probability mass  $m(i, E_{Nr})$  also represents the degree of belief in state  $s_i$  after considering all retrieved cases. Hence the combined degrees of belief are directly given by

$$\beta_i = m(i, E_{Nr}) \quad i = 1 \cdots n \quad (13a)$$

$$\beta_S = m(S, E_{Nr}) = 1 - \sum_{i=1}^n \beta_i \quad (13b)$$

where  $\beta_S$  refers to the degree of belief unassigned to any individual state after all retrieved cases have been incorporated. It indicates a degree of ignorance or incompleteness of information in the generated assessment.

Further, from the plausibility definition in (4), the value of plausibility for the hypothesis that state  $s_i$  is true is equal to  $\beta_i + \beta_S$ . It is the upper bound of the likelihood for the truth of state  $s_i$ . The lower bound for the likelihood of state  $s_i$  is reflected by the belief degree  $\beta_i$ . In other words, we obtain the interval  $[\beta_i, \beta_i + \beta_S]$  as estimate of the probability for state  $s_i$  ( $i=1 \dots n$ ) by using the D-S combination rule. In the next subsection we shall discuss how to refine these initial estimates to obtain crisp probability values of states by doing probabilistic reasoning.

### 3.2 Reaching Final Probabilities via Probabilistic Reasoning

The probability intervals derived from the D-S rule can be refined via probabilistic reasoning. Without any prior knowledge, the initial probability  $P_0(s_i)$  for a state is defined by equally distributing the unassigned probability  $\beta_S$  among all states. Thus we have

$$P_0(s_i) = \beta_i + \frac{\beta_S}{n} \quad i = 1 \cdots n \quad (14)$$

Then we perform probability updating based on the Bayes theorem.

As cases in the case base were collected independently of each other, we utilized the similar relation with every retrieved case,  $sr(C_j)$ , as an independent observation to update the prior probabilities according the Bayes theorem. Define  $H_j$  as the set of the first  $j$  observations (similar relations) as follows:

$$H_j = \{sr(C_1), sr(C_2), \dots, sr(C_j)\} \tag{15}$$

The Bayesian reasoning for probabilities of states can be summarized in a recursive form by

$$\begin{aligned} P(s_i | H_{j+1}) &= \frac{P(s_i | H_j) \cdot P(sr(C_{j+1}) | H_j, s_i)}{\sum_{k=1}^n P(sr(C_{j+1}) | H_j, s_k) \cdot P(s_k | H_j)} \\ &= \frac{P(s_i | H_j) \cdot P(sr(C_{j+1}) | s_i)}{\sum_{k=1}^n P(sr(C_{j+1}) | s_k) \cdot P(s_k | H_j)} \quad j = 0, \dots, Nr - 1 \end{aligned} \tag{16}$$

Note that we have  $P(s_i | H_0) = P_0(s_i)$  to start with this recursive form.

It can be seen from Eq. (16) that, to update probabilities of states, we need the conditional probability  $P(sr(C_j)|s_i)$  for all the retrieved cases  $C_j$  ( $j=1\dots Nr$ ). Such probability can be regarded as the likelihood of a randomly picked case from the case base being similar to  $C_j$  provided that the state in this case is known as  $s_i$ . Hence we have

$$P(sr(C_j) | s_i) = \sum_{\substack{C \in \text{Case Base} \\ C \text{ similar to } C_j}} P(C | s_i) \quad i = 1 \dots n, \quad j = 1 \dots Nr \tag{17}$$

Further, we apply the Bayes theorem and transform the probability  $P(C|s_i)$  to the following form:

$$P(C | s_i) = \frac{P(C) \cdot P(s_i | C)}{\sum_{C_k \in \text{Case base}} P(s_i | C_k) \cdot P(C_k)} \quad i = 1 \dots n \tag{18}$$

Since we assume all cases in the case library are equally probable to be selected, Eq. (18) is simplified to

$$P(C | s_i) = \frac{P(s_i | C)}{\sum_{C_k \in \text{Case base}} P(s_i | C_k)} = \frac{P(s_i | C)}{\sum_{\forall k} P_k(s_i)} \quad i = 1 \dots n \tag{19}$$

At this point, it has been obvious that we can calculate the probability  $P(C|s_i)$  by using probabilistic information stored in individual cases in the case library, which further enables updating state probabilities in terms of Eqs. (16) and (17).

However, one disadvantage of the above calculation with Bayes theorem is that different similarities (thereby importances) of the cases are not taken into account. For more accurate results, we are not directly adopting such assessment as final probability values. Instead we utilize the probability values yielded from Bayesian reasoning as factors to divide the unassigned probability  $\beta_S$  across various states. This means that every state  $s_i$  receives an additional probability mass from  $\beta_S$  in proportion

to  $P(s_i|H_{Nr})$ . This additional mass is then added to the lower bound of probability,  $\beta_i$ , to settle the final probability assessment. In other words, after information fusion in two steps, the probability for state  $s_i$  is finalized as

$$P(s_i) = \beta_i + P(s_i | H_{Nr}) \cdot \beta_s \quad i = 1 \cdots n \tag{20}$$

### 4 Derivation of General Utilities of Actions Given States

The basic idea is to derive the general utility of performing one action under a given state by using information from the case library. However, owing to the fact that no exact information is known about states in cases, case specific utilities recorded can not provide direct answers to our inquiries. As an alternative, we here attempt to estimate this utility with an expected value by considering all those cases in which the underlying action was performed. By  $Sub(a)$  we denote the subset of cases in the case library in which the action  $a$  was performed. Then the expected value of the general utility of performing action  $a$  given state  $s_i$  can be given by:

$$U(a | s_i) = \sum_{C_t \in Sub(a)} U(a | C_t) \cdot P_a(C_t | s_i) \tag{21}$$

As  $U(a|C_t)$  represents the known utility recorded in case  $C_t$ , what remains to resolve is the probability  $P_a(C_t|s_i)$ . By employing the Bayes theorem, this probability is reformulated as

$$P_a(C_t | s_i) = \frac{P_a(C_t) \cdot P(s_i | C_t)}{\sum_{C_k \in Sub(a)} P_a(C_k) \cdot P(s_i | C_k)} \tag{22}$$

Considering that cases in the subset  $Sub(a)$  are equally probable to be picked up, Eq. (22) is reduced to

$$P_a(C_t | s_i) = \frac{P(s_i | C_t)}{\sum_{C_k \in Sub(a)} P(s_i | C_k)} \tag{23}$$

Since  $P(s_i|C_k)$  is available as the probability of state  $s_i$  in case  $C_k$ , we easily resolve Eq. (23), leading to computation of the expected value of the general utility according to Eq. (21). This expected value then enters the decision tree as estimation of the (general) utility of action  $a$  given state  $s_i$ .

### 5 Decision Analysis Using Case-Based Decision Model

Once a decision model is constructed from cases, it can be applied to analyse and evaluate alternative actions in the current situation, taking into account both likelihoods and probable consequences. We will first introduce a well established principle for doing such analysis of decisions, followed by discussions of how this

basic principle can be applied in circumstances when utility values specified in individual cases are fuzzy or imprecise.

**5.1 Principle of Maximizing Expected Utility**

With complete information in the decision tree derived, we can now compute the expected utility of the various alternative actions. The expected utility of action  $a_j$  is defined as

$$EU(a_j) = P(s_1) \cdot U(a_j | s_1) + P(s_2) \cdot U(a_j | s_2) + \dots + P(s_n) \cdot U(a_j | s_n) \quad (24)$$

where  $P(s_i)$  and  $U(a_j|s_i)$  represent the probability and (general) utility values derived from the retrieved cases and the case library respectively. Then a choice should be made among the alternatives according to the principle of maximizing the expected utility [2], which is formulated as follows:

**The principle of maximizing expected utility (MEU):** In a given decision situation the deciding agent should prefer the alternative with maximal expected utility. That means that alternative  $a_1$  is preferred to  $a_2$  if and only if  $EU(a_1) > EU(a_2)$ .

The expected utility of an action approximates the mean utility score that will be obtained if an agent or decision maker meets the situation many times and chooses and conducts the same action constantly. In view of this, the significance of the MEU principle is to optimize the long term performance of decision making under uncertainty.

The merit of doing decision analysis after CBR can be illustrated with the following example. Assume that, given a target situation, two cases  $C_1$  and  $C_2$  are retrieved from the case base and they have actions  $a_1$  and  $a_2$  respectively. Both cases are assigned with good utility values as evaluations of their outcomes, but case  $C_1$  is more similar to the target situation. Then, according to CBR alone, action  $a_1$  associated with case  $C_1$  will be judged more suitable as solution to the new situation. Nevertheless, if we further consider more information in the decision tree, we might change our preference after decision analysis.

For instance, suppose that the state probabilities and utilities of actions under possible states ( $s_1$  and  $s_2$ ) are derived from previous cases as follows:

$$\begin{array}{lll} P(s_1 | sr(C_1), sr(C_2)) = 0.6 & U(a_1 | s_1) = 70 & U(a_2 | s_1) = 40 \\ P(s_2 | sr(C_1), sr(C_2)) = 0.4 & U(a_1 | s_2) = -90 & U(a_2 | s_2) = 60 \end{array}$$

The expected utilities of  $a_1$  and  $a_2$  are calculated as  $EU(a_1)=6$  and  $EU(a_2)=48$  respectively in the current situation. Hence we will prefer action  $a_2$  according to the MEU principle. We believe that  $a_2$  is a more rational choice considering the high risk of action  $a_1$  under state  $s_2$ . This rational choice is achieved by taking advantage of the case-based decision tree which accommodates more information than case similarity alone.

**5.2 When Utility Values from Cases Are Fuzzy**

Until now we have assumed that a crisp utility value is assigned in every case in the case library. However, in numerous practical applications, it is frequently difficult for



human users and even domain experts to assign an exact utility as their evaluation of the consequences. They would be more likely to say that the outcome in a case should receive a utility score of, say, around 60. Here “around 60” is an imprecise value and, in terms of fuzzy set theory [23], it is considered as a fuzzy number. Our intention is to extend the representation to allow for users to specify vague, fuzzy data as evaluations of utility in specific cases. But, by doing this, we will not exclude the possibility that users assign crisp utility values if they prefer. Considering that crisp numbers are special singleton fuzzy numbers, this extension would bring a useful generalization of the theory and methods making our framework applicable to more general types of data and information.

In fuzzy set theory, a fuzzy number is a fuzzy subset of  $R$  that is convex and normal. So, in principle, users can define any convex and normal fuzzy subset of  $R$  as fuzzy utility in a specific case. But, for reducing computational complexity, we would prefer to recommend triangular fuzzy numbers which are intuitive, simple, and easy to manipulate. A triangular fuzzy number  $F$  can be depicted by a 3-tuple:  $F = (f_1, f_2, f_3)$ , with its membership function being illustrated in Fig. 4.

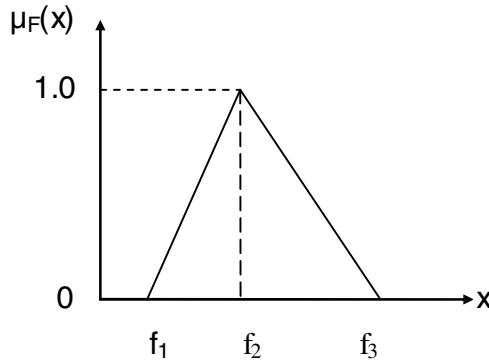


Fig. 4. A triangular fuzzy number

Two nice properties of triangular fuzzy numbers are that the addition of two triangular fuzzy numbers is still a triangular fuzzy number and that the multiplication of a constant with a triangular fuzzy number is still a triangular fuzzy number [24]. That is to say that, given two triangular fuzzy numbers  $F = (f_1, f_2, f_3)$ , and  $G = (g_1, g_2, g_3)$ , and a constant  $\gamma$ , we have

$$F + G = (f_1 + g_1, f_2 + g_2, f_3 + g_3) \tag{25}$$

$$\gamma \cdot F = (\gamma \cdot f_1, \gamma \cdot f_2, \gamma \cdot f_3) \tag{26}$$

Owing to the properties depicted in Eqs. (25) and (26), we clearly see that the general utility estimation in (21) and the expected utility of actions in (24) are also triangular fuzzy numbers as long as utilities in individual cases are specified as triangular fuzzy numbers. Consequently, evaluating alternatives according to the EMU principle turns to studying the fuzzy dominance relations between the fuzzy expected utilities

represented as fuzzy numbers. This task is not trivial in the sense that a natural order does not exist with the quantities as fuzzy numbers.

Let  $F_i$  and  $F_j$  be two fuzzy numbers corresponding to the fuzzy expected utilities for alternative actions  $a_i$  and  $a_j$  respectively. The fuzzy relation that  $a_i$  is dominated by  $a_j$  (or  $F_i$  is dominated by  $F_j$ ) is defined by the degree of possibility and the degree of necessity of the event  $F_i < F_j$ , which are given by

$$\Pi(F_i < F_j) = \sup_{x < y} \min(\mu_{F_i}(x), \mu_{F_j}(y)) \tag{27}$$

$$N(F_i < F_j) = 1 - \sup_{x \geq y} \min(\mu_{F_i}(x), \mu_{F_j}(y)) \tag{28}$$

Further, we investigate to what extent an action is a dominated one. Since the statement that  $a_i$  is dominated becomes true if  $a_i$  is dominated by at least one of the other alternatives, we apply an  $s$ -norm as logical disjunction to connect the dominance relations between  $a_i$  and the others. If the maximum operator is adopted as the means for  $s$ -norm, the degrees of possibility and necessity of alternative  $a_i$  being dominated are respectively defined as

$$Poss(a_i) = \max_{i \neq j} (\Pi(F_i < F_j)) \tag{29}$$

$$Nec(a_i) = \max_{i \neq j} (N(F_i < F_j)) \tag{30}$$

By means of the possibility and necessity values given in (29) and (30), we actually have defined a fuzzy subset of dominated alternative actions. Finally, we define an  $\alpha$ - $\beta$ -cut of this fuzzy subset to reach a crisp subset  $DOM$ . The membership function of the crisp subset  $DOM$  is given by

$$\mu_{DOM}(a_i) = \begin{cases} 1 & Poss(a_i) \geq \alpha \text{ and } Nec(a_i) \geq \beta \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

where  $\alpha, \beta$  ( $0 < \beta < \alpha < 1$ ) are parameters controlling the number of dominated actions. The remaining alternatives are subsequently recommended to the decision maker as non-dominated solutions.

## 6 Related Work

A probabilistic model for CBR was first proposed in [25]. The basic idea presented there is to consider the CBR principle that similar problems have similar solutions as a “rule of thumb” rather than a universally valid rule. According to this probabilistic model, the conventional CBR principle can be reformulated into a heuristic rule stating that similar problems are at most likely to have similar solutions. The merit of this formulation is that it allows for exceptions to the CBR rule.

Later, a similarity-based inference scheme [26] was developed from the CBR probabilistic model [25] by the same author. The method is to represent information from relevant cases into belief functions for characterizing confidence of alternative solutions for a new problem at hand. Then the belief functions from individual cases

are combined in the framework of information fusion. This method can be useful in the overall problem solving process by measuring different confidence levels of different candidate solutions.

We proposed a framework for case-based decision analysis in [27], in which the probabilistic information from individual cases was integrated solely with the Bayes theorem. The weakness is that it can not take into account the different degrees of similarity of retrieved cases in probabilistic calculation. This problem is overcome with the work presented here by using the D-S theory for evidential combination with respect to cases. Our work differs from [25] and [26] in that it does not directly evaluate solutions and their confidences. Instead it aims to produce an intermediate decision model by fusing information from cases to highlight all possibilities and consequences. Decision analysis can then be conducted on the derived decision model to identify the most promising solution in view of expected outcomes.

## 7 Conclusion

This paper presents a new framework for case-based decision analysis supported by CBR. We claim that CBR and decision theory can complement each other in a coherent, hybrid system. CBR has the strength of creating a situation dependent decision model without domain knowledge. This is achieved by deriving states probabilities and general utility estimates from previous cases through an information fusion process comprising evidence combination and probabilistic reasoning. It follows that more accurate and objective data will be available in the decision model, promoting more reliable results of decision analysis. On the other hand, decision theory helps CBR better tackling the uncertainty issue by considering all probable consequences, risks, and likelihoods rather than similarity of cases alone. This would endow the agent or decision maker with more complete awareness of the situation and environment for making predictive, secured and rational choices. Besides, we have shown that fuzzy numbers can be used to represent case specific utility values for decision analysis and that fuzzy and probabilistic information are well utilized together in our case-based framework.

In future we will apply our approach to support decision making in strategic maintenance scenarios in industry. Therein a machine or production line under investigation can be considered as the environment, and evaluation grades or faults of the machine refer to the internal states of the environment. We plan to not only assess probable grades for machines but also carry out decision analysis based on previous cases to find out effective, rational, low risk counter-measures (maintenance plans, repair alternatives, etc.) as decision support.

## References

1. Weiss, G.: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge (1999)
2. Gärdenfors, P., Sahlin, N.E.: *Introduction: Bayesian Decision Theory – Foundations and Problems*. In: *Decision, Probability, and Utility*, pp. 1–15. Cambridge University Press, Cambridge (1997)

3. Raiffa, H.: *Decision Analysis: Introductory Readings on Choices under Uncertainty*. McGraw Hill, New York (1997)
4. Gilboa, I., Schmeidler, D.: Case-Based Decision Theory. *The Quarterly Journal of Economics* 110, 605–639 (1995)
5. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Com.* 7, 39–59 (1994)
6. Doyle, D., Cunningham, P., Bridge, D., Rahman, Y.: Explanation Oriented Retrieval. In: Funk, P., González Calero, P.A. (eds.) *ECCBR 2004*. LNCS, vol. 3155, pp. 157–168. Springer, Heidelberg (2004)
7. Roth-Berghofer, T.R.: Explanations and Case-Based Reasoning: Foundational Issues. In: Funk, P., González Calero, P.A. (eds.) *ECCBR 2004*. LNCS, vol. 3155, pp. 389–403. Springer, Heidelberg (2004)
8. Brinker, K., Hullermeier, E.: Label Ranking in Case-Based Reasoning. In: Weber, R.O., Richter, M.M. (eds.) *ICCBR 2007*. LNCS, vol. 4626, pp. 77–91. Springer, Heidelberg (2007)
9. Coyle, L., Cunningham, P.: Improving Recommendation Ranking by Learning Personal Feature Weights. In: Funk, P., González Calero, P.A. (eds.) *ECCBR 2004*. LNCS, vol. 3155, pp. 560–572. Springer, Heidelberg (2004)
10. McSherry, D.: Completeness Criteria for Retrieval in Recommender Systems. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 9–29. Springer, Heidelberg (2006)
11. Nicholson, R., Bridge, D., Wilson, N.: Decision Diagrams: Fast and Flexible Support for Case Retrieval and Recommendation. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 136–150. Springer, Heidelberg (2006)
12. Stahl, A.: Combining Case-Based and Similarity-Based Product Recommendation. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 355–369. Springer, Heidelberg (2006)
13. Aamodt, A.: CBR for Advice Giving in a Data-Intensive Environment. In: *Proceedings of 10th Scandinavian Conference on Artificial Intelligence*, Stockholm, pp. 201–205 (2008)
14. Ekenberg, L., Boman, M., Linneroth-Bayer, J.: General Risk Constraints. *Journal of Risk Research* 4, 31–47 (2001)
15. Shapira, Z.: *Risk Taking: A Managerial Perspective*. Russel Sage Foundation, Thousand Oaks (1995)
16. Stahl, A., Gabel, T.: Using Evolution Programs to Learn Local Similarity Measures. In: Ashley, K.D., Bridge, D.G. (eds.) *ICCBR 2003*. LNCS, vol. 2689, pp. 537–551. Springer, Heidelberg (2003)
17. Cheng, W., Hullermeier, E.: Learning Similarity Functions from Qualitative Feedback. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) *ECCBR 2008*. LNCS, vol. 5239, pp. 120–134. Springer, Heidelberg (2008)
18. Gabel, T., Riedmiller, M.: Increasing Precision of Credible Case-Based Inference. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) *ECCBR 2008*. LNCS, vol. 5239, pp. 225–239. Springer, Heidelberg (2008)
19. Xiong, N., Funk, P.: Learning Similarity Metric Reflecting Utility in Case-Based Reasoning. *Journal of Intelligent and Fuzzy Systems* 17, 407–416 (2006)
20. Shafer, G.: *A Mathematical Theory of Evidence*. Princeton University Press, Princeton (1976)
21. Smets, P.: Belief Functions. In: Smets, P., Mamdani, E.H., Dubois, D., Prade, H. (eds.) *Non-Standard Logics for Automated Reasoning*, pp. 253–277. Academic Press, London (1988)

22. Dempster, A.P.: Upper and Lower Probabilities Induced by a Multi-Valued Mapping. *Ann. Math. Stat.* 38, 325–339 (1967)
23. Zadeh, L.A.: The Concept of a Linguistic Variable and Its Applications to Approximate Reasoning. *Information Sciences* 8, 199–249 (1975)
24. Kaufmann, A., Gupta, M.: *Introduction to Fuzzy Arithmetic. Theory and Applications.* Van Nostrand Reinhold Company Inc. (1985)
25. Hullermeier, E.: Toward a Probabilistic Formalization of Case-Based Inference. In: *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 248–253 (1999)
26. Hullermeier, E.: Similarity-Based Inferences as Evidential Reasoning. In: *Proceedings of 14th European Conference on Artificial Intelligence*, pp. 55–59 (2000)
27. Xiong, N., Funk, P.: A novel framework for case-based decision analysis. In: *Proceedings of 10th Scandinavian Conference on Artificial Intelligence*, Stockholm, Sweden, pp. 141–148 (2008)

# Constraint-Based Case-Based Planning Using Weighted MAX-SAT

Hankui Zhuo<sup>1</sup>, Qiang Yang<sup>2</sup>, and Lei Li<sup>1</sup>

<sup>1</sup> Software Research Institute, Sun Yat-sen University, Guangzhou, China  
zhuohank@gmail.com, lnsllilei@mail.sysu.edu.cn

<sup>2</sup> Computer Science and Engineering, Hong Kong University of Science and Technology,  
Clearwater Bay, Kowloon, Hong Kong  
qyang@cse.ust.hk

**Abstract.** Previous approaches to case-based planning often finds a similar plan case to a new planning problem to adapt to solve the new problem. However, in the case base, there may be some other cases that provide helpful knowledge in building the new solution plan. Likewise, from each existing case there may be only certain parts that can be adapted for solving the new problem. In this paper, we propose a novel constraint-based case-based planning framework that can consider all similar plans in a case base to the current problem, and take only portions of their solutions in adaptation. Our solution is to convert all similar plan cases to constraints, and use them to solve the current problem by maximally exploiting the reusable knowledge from all the similar plan cases using a weighted MAX-SAT solver. We first encode a new planning problem as a satisfiability problem, and then extract constraints from plan cases. After that, we solve the SAT problem, including the extracted constraints, using a weighted MAX-SAT solver and convert the solution to a plan to solve the new planning problem. In our experiments, we test our algorithm in three different domains from International Planning Competition (IPC) to demonstrate the efficiency and effectiveness of our approach.

## 1 Introduction

Automatic planning aims to find an action sequence that transforms an initial state to a goal state. Researchers have built different algorithms to solve the problem efficiently. Classical planning involves the generation of plans by state or partial plan space search in order to satisfy a given goal [16,17]. Because planning is often difficult to do, case-based Reasoning has been introduced as a general problem-solving paradigm that makes use of the notion of analogy. Case-based reasoning uses domain-specific knowledge of previously experienced, concrete problem solutions in order to solve a new problem. It accomplishes this task by finding a similar past case and reusing it in the new problem situation. Part of its feasibility is founded on psychological studies, where it is found that humans often solve new problems by analogy. Several studies have given empirical evidence for the dominating role of specific, previously experienced situations in human problem solving.

In previous case-based planning approaches, one way is to build a plan by transformational analogy which is a problem-solving technique in which a pre-selected plan,

defined as a sequence of actions, is modified to solve a new problem [20]. Possible modifications to the plan include removing actions, adding new actions, and changing the parameters from actions. The CHEF system constructs cooking recipes, which are plans because recipes are sequences of cooking steps such as boiling a certain amount of water [7]. These recipes are modified depending on factors such as the ingredients currently available. Interest on case-based planning has revived recently, e.g., in the work of [5].

The previous approaches on case-based planning mainly focus on adapting *a single* similar plan case in its *entirety*, which is found from the set of plan cases, to a new one that solve a new planning problem. However, there may be some other cases in the same case base that can provide some helpful knowledge in building the new plan, although they may not be so similar to the new problem. Similarly, it may not be the case where a plan is completely reusable; instead, each case that is being reused may only partially contribute to the new solution. Thus, it is important for us to develop a new method to exploit all the partially helpful information to help build a new plan, in order to improve the planning efficiency and improve its effectiveness.

In this paper, we propose a novel case-based planning framework called MAXCBP, which stands for *using a weighted MAX-SAT solver to do Cases-Based Planning*. In this algorithm, we extract the useful information from all the similar plan cases in the form of constraints. We also formulate the new planning problem as a constraint via a procedure similar to Graphplan[6]. We add the two sets of constraints to help search for a plan for a new planning problem.

In particular, our MAXCBP algorithm works in the following three steps. First, we encode the planning problem as a set of clauses (a satisfaction problem). Secondly, we extract constraints that can be also converted as a set of clauses, from all the plan cases. Finally, we assign weights to all the clauses and solve them with a weighted MAX-SAT solver. We then convert the final solution to a plan solution to the new planning problem. Compared to previous approaches, our algorithm has the advantage that it can selectively use partial knowledge from each case solution as needed to solve a new problem. In addition, we make use of all available cases in the case base to solve a new problem.

The rest of the paper is organized as follows. We first give some related work to this paper, and then address our problem definition. After that, we describe the detailed steps of our algorithm. In the experiment section, we evaluate our algorithm in three planning domains. Finally, we conclude the paper and discuss future work.

## 2 Related Work

### 2.1 Case-Based Planning

Case-based planning (CBP) uses a set of cases that consist of a past problem, a goal and a plan that makes a transition from the problem to the goal [7]. Given a new problem, CBP systems retrieve one or more cases whose problem is similar to the current one and adapt the plans contained in the retrieved cases to achieve the new goal. Case retrieval involves an intelligent search among all cases to find the ones that are adequate to solve the new problem. The result of the plan adaptation process is a solution that includes parts that are derived from selected case and new parts derived by first-principle based planning. [5] analyzes the current state of art in case-based plan adaptation research. This

work presents six dimensions for categorizing various aspects of existing case-based plan adaptation algorithms, including the type of transformation, the role of the case content, the use of case merging, the representation formalism, and the computational complexity of the algorithm. It uses these dimensions as a framework to compare various systems.

[11] presents a scheme for learning the case quality based on its utility during a validation phase. The quality obtained determine the way in which these cases are preferred in the retrieval and replay processes. It shows that the planning performance can be improved when case utilities are used. [9] proposes a general framework for transformational analogy. It demonstrates that transformational analogy does not meet a crucial condition for a well-known worst-case complexity scenario, and that plan adaptation can be computationally harder than planning from the scratch, is not applicable for transformational analogy. [8] develops an on-line case-based planning framework. In this framework, when a plan is retrieved, a plan dependency graph is inferred to capture the relations between actions in the plan. Then the plan is adapted in real-time using its plan dependency graph, which allows the system to create and adapt plans in an efficient and effective manner while performing the task.

## 2.2 Planning as Satisfiability and Weighted MAX-SAT

[2] develops a formal model of planning based on satisfiability. The satisfiability approach uses the best-known logical formalization of planning, based on the situation calculus [4]. In this system, the execution of an action is explicitly represented by the application of a function to a term representing the state in which the action is performed. The satisfiability approach provides not only a more flexible framework for stating different kinds of constraints on plans, but also a more accurate theory behind modern constraint-based planning systems. [3] describes a two-phase algorithm for MAX-SAT and weighted MAX-SAT problems. Firstly, it uses the GSAT heuristic to find a good solution to the problem. Secondly, it uses an enumeration procedure based on the Davis-Putnam-Loveland algorithm to find a provably optimal solution.

## 2.3 Learning for Planning

Planning problems are often formulated as heuristic search and the choice of the heuristic function plays a significant role in the performance of planning systems. [12] proposes an approach to learning heuristic functions from previously solved problem instances in a given domain. The approach is based on approximate linear programming, which is commonly used in reinforcement learning. [13] presents a novel approach for boosting the scalability of heuristic planners based on automatically learning domain-specific search control knowledge from planning contexts in the form of relational decision trees. The contexts are defined as the set of helpful actions extracted from the relaxed planning graph of a given state, the goals remaining to be achieved, and the static predicates of the planning task. [14] introduces a novel feature space for representing control knowledge. It defines features in terms of information computed via relaxed plan extraction, which has been a major source of success for non-learning planners, which gives a new way of leveraging relaxed planning techniques in the



context of learning. The authors show that the approach is able to surpass state-of-the-art non-learning planners across a wide range of planning competition domains.

### 3 Problem Formulation

In this work, we consider the restrained form of STRIPS model [15], leaving more complex models such as ADL as our further work. A planning domain is defined in this work as  $\Sigma = (S, A, \gamma)$ , where  $S$  is the set of states,  $A$  is the set of action models,  $\gamma$  is the deterministic transition function  $S \times A \rightarrow S$ . Each action model in  $A$  is composed of three parts: an action name with zero or more arguments, a set of preconditions which should be satisfied before the action is executed, and a set of effects which are the results of executing the action. A planning problem can be defined as  $\mathcal{P} = (\Sigma, s_0, g)$ , where  $s_0$  is an initial state, and  $g$  is a goal state. A solution to a planning problem is an action sequence  $(a_0, a_1, \dots, a_n)$  called a plan, which makes a projection from  $s_0$  to  $g$ . Each  $a_i$  is an *action schema* composed of an action name and zero or more arguments. Furthermore, a *plan case* is defined as  $T = (s_0, a_0, a_1, \dots, a_n, g)$ . Notice that the intermediate states between actions can be computed using the action models. In this paper, we denote a set of *plan cases* as  $PC$ .

Our problem can be formulated as follows. We are given a planning domain  $\Sigma$ , a set of *plan cases*  $PC$  and a new plan problem  $P$ . Our algorithm MAXCBP outputs a plan for the new plan problem  $P$  using the helpful information from  $PC$ . An example problem description is given in Table 1, which is from the domain of *blocks*.

Table 1. An example problem description

input: the domain of <i>blocks</i>	
predicates	action models
(on ?x - block ?y - block)	(pick-up ?x - block)
(ontable ?x - block)	preconditions: (clear ?x)(ontable ?x)(handempty)
(clear ?x - block)	effects: (holding ?x)(not(ontable ?x))(not(clear ?x))(not(handempty))
(handempty)	(put-down ?x - block) ( <b>preconditions &amp; effects omitted</b> )
(holding ?x - block)	(stack ?x - block ?y - block) ( <b>preconditions &amp; effects omitted</b> )
	(unstack ?x - block ?y - block) ( <b>preconditions &amp; effects omitted</b> )
input: plan cases	
plan case 1: (ontable A) (ontable B) (clear A) (clear B) (handempty), (pick-up A) (stack A B), (handempty) (on A B)	
plan case 2: (ontable B) (on A B) (clear A) (handempty), (unstack A B) (put-down A) (pick-up B) (stack B A), (on B A) (ontable A)	
...	
input: a new planning problem	
initial state $s_0$	goal $g$
(ontable A) (clear A)	(on A B)
(holding B)	(ontable B)
output: a plan for the new planning problem	
(put-down B) (pick-up A) (stack A B)	

<sup>1</sup> <http://www.cs.toronto.edu/aips2000/>

## 4 The MAXCBP Algorithm

Our solution is to first encode the new planning problem into a constraint satisfaction problem. The idea similar to Graphplan [6], where we consider  $K$  steps of actions in which to solve a problem. When we solve the problem using a constraint satisfaction problem formulation, the solution may be time-consuming. Thus, we make use of the knowledge from the case base  $PC$ , which we convert to another set of constraints. The two sets of constraints are combined into a new satisfiability problem, which is solved using a MAXSAT solver.

In the following, we first give an overview of our algorithm MAXCBP. A detailed description of each steps will be given in sections 4.1-4.4.

### 4.1 Encoding Planning Problem

In step 1 of Algorithm 1 we encode a planning problem  $\mathcal{P}$  as a satisfiability problem [2], which is simply a set of axioms with the property that any model of the axioms corresponds to a valid plan. Some of these axioms describe the initial  $s_0$  and goal states  $g$ . For the example in Table 1  $s_0$  and  $g$  can be described as

$$(ontable\ A\ 1) \wedge (clear\ A\ 1) \wedge (holding\ B\ 1) \wedge (on\ A\ B\ \infty) \wedge (ontable\ B\ \infty)$$

The other axioms describe the actions in general, which include the standard effect and frame axioms, plus others that rule out the anomalous models.

First, we rule out the possibility that an action executes despite the fact that its preconditions are false. This can be done by asserting that an action implies its preconditions as well as effects; e.g., for preconditions of the action *pick-up* in Table 1

$$\forall x, i. (pick-up\ x\ i) \rightarrow (clear\ x\ i) \wedge (ontable\ x\ i) \wedge (handempty\ i)$$

and for effects,

$$\forall x, i. (pick-up\ x\ i) \rightarrow (holding\ x\ i+1) \wedge \neg(clear\ x\ i+1) \\ \wedge \neg(ontable\ x\ i+1) \wedge \neg(handempty\ i+1)$$

---

#### Algorithm 1. Overview of our MAXCBP algorithm

---

**Input:** a new planning problem  $\mathcal{P} = (\Sigma, s_0, g)$ , a set of plan cases  $PC$ ;

**Output:** a plan  $P$  for the planning problem  $\mathcal{P}$ ;

- 1: we first consider a plan graph according to Graphplan [6], where the plan is of length  $k$  ( $k$  is set to one initially, and incremented by one if no solution is found). We encode the planning problem  $\mathcal{P}$  as a set of clauses  $C$  (a satisfaction problem) for a solution plan of up to length  $k$ ;
  - 2: build constraints C1-C5 from  $PC$ ;
  - 3: assign each constraint of C1-C5 with its appearing frequency as its weight;
  - 4: assign each clause in  $C$  with a high weight that the clause should hold;
  - 5: solve all the weighted constraints with a weighted MAXSAT solver;
  - 6: if no solution is found, we increment  $k$  by one, and go to step 1.
  - 7: convert the solved result to a plan  $P$ ;
  - 8: **return**  $P$ ;
-

It is interesting to note that in this formulation preconditions and effects are treated symmetrically.

Next, we state that only one action occurs at a time, e.g.,

$$\forall x, x', i. (x \neq x') \rightarrow \neg(\text{pick-up } x \ i) \vee \neg(\text{pick-up } x' \ i)$$

Finally, we assert that some action occurs at every time step. This is not a significant restriction, since we can always introduce an explicit “do nothing” action if desired. For the action *pick-up* in Table 1 the axiom schema is

$$\forall i < N. \exists x. (\text{pick-up } x \ i)$$

(An existentially-quantified formula expands to the disjunction of its instantiations.)

If a planning problem is specified by asserting a complete initial state then these axioms guarantee that all models correspond to valid plans. This is so because every model contains a sequence of actions whose preconditions are satisfied, and the execution of an action in a state completely determines the truth-values of all propositions in the next state. The only model of the simple two step planning problem is the intended model containing (*put-down* B 1), (*pick-up* A 2) and (*stack* A B 3). A simple planning system can be constructed by linking a routine that instantiates such a given set of axiom schemas and initial and goal state specifications to a Boolean satisfiability algorithm.

## 4.2 Building Constraints

In step 2, we wish to build constraints to represent the relationship between actions in *PC*. We observe that there are five kinds of relationships between each two actions in a plan case, i.e.,

1. one action provides a precondition for its subsequent action (we call this relationship as an *add-pre constraint*);
2. one action adds an effect but deleted by its subsequent action (we call this relationship as an *add-del constraint*);
3. one action deletes an effect but added by its subsequent action (we call this relationship as a *del-add constraint*);
4. one action shares a precondition with its subsequent action (we call this relationship as a *pre-pre constraint*);
5. a precondition of one action is deleted by its subsequent action (we call this relationship as a *pre-del constraint*).

We denote a plan case  $pc \in PC$  as  $pc = (a_1, a_2, \dots, a_n)$ , each action pair in  $pc$  as  $\langle a_i, a_j \rangle$  where  $1 \leq i < j \leq n$ . We formulate the idea as follows.

**C1: *add-pre constraints*.** For each action pair  $\langle a_i, a_j \rangle$ , the idea that  $a_i$  provides a precondition for  $a_j$  is, there is a proposition  $p$  which is added by  $a_i$  and used as a precondition of  $a_j$ . We denote a list of effects added by  $a_i$  as  $add_i$ , and a list of preconditions of  $a_j$  as  $pre_j$ . Then, we can formulate this constraint as follows.

$$p \in add_i \wedge p \in pre_j$$

where parameters of  $p$  are included by  $a_i$  and  $a_j$ . Intuitively, the action  $a_j$  requires that  $a_i$  should be executed first in a plan, that  $a_j$  can be executed and produce some useful effects.

**C2: add-del constraints.** For each action pair  $\langle a_i, a_j \rangle$ , the idea that  $a_i$  adds an effect but deleted by  $a_j$  is, there is a proposition  $p$  which is added by  $a_i$  and used as a precondition of  $a_j$ . We denote a list of effects deleted by  $a_j$  as  $del_j$ . Then, we can formulate this constraint as follows.

$$p \in add_i \wedge p \in del_j$$

Intuitively,  $a_j$  needs to be executed to deleted a redundantly added effect by  $a_i$ , that no unnecessary actions will be executed after  $a_j$  in a plan.

**C3: del-add constraints.** For each action pair  $\langle a_i, a_j \rangle$ , there is a proposition  $p$  which is deleted by  $a_i$  and added by  $a_j$ . Then, this constraint can be formulated by

$$p \in del_i \wedge p \in add_j$$

This constraint specifies that,  $a_j$  is needed to add an effect which is unexpectedly deleted by  $a_i$ .

**C4: pre-pre constraints.** For each action pair  $\langle a_i, a_j \rangle$ , there is a proposition  $p$  which is a precondition of  $a_i$  and  $a_j$ . Then, this constraint can be formulated by

$$p \in pre_i \wedge p \in pre_j$$

This constraint specifies that different actions may be executed together under the same preconditions, e.g., frame axioms which are not changed between these actions in a plan.

**C5: pre-del constraints.** For each action pair  $\langle a_i, a_j \rangle$ , there is a proposition  $p$  which is a precondition of  $a_i$  but deleted by  $a_j$ . Then, this constraint can be formulated by

$$p \in pre_i \wedge p \in del_j$$

This constraint specifies that  $a_j$  should be executed to delete  $p$  that actions with the precondition  $p$  will not be executed again in a plan.

With respect to the restrained form of STRIPS model, we assert that the above five kinds of constraints encode all the possible relationship between actions. Before giving proof to this conclusion, we provide the following two requirements according to the restrained form of STRIPS model, i.e.,

**R1:** A proposition  $p$  added by action  $a_i$  should not be a precondition of  $a_i$ , i.e.,

$$p \in add_i \rightarrow p \notin preState_i$$

where  $preState_i$  is a list of propositions that exist before  $a_i$  is executed. Notice that  $pre_i \subseteq preState_i$ .

**R2:** A proposition  $p$  deleted by action  $a_i$  should be a precondition of  $a_i$ , i.e.,

$$p \in del_i \rightarrow p \in pre_i$$

Then, we have the theorem under the conditions of R1-R2, as shown in the following. Notice that, when we consider a proposition referring to an action pair  $\langle a_i, a_j \rangle$ , we assume that there is no other actions between  $a_i$  and  $a_j$  that affect the proposition.

**Theorem:** constraints C1-C5 encode all the possible relationships between two actions of  $\langle a_i, a_j \rangle$  in a plan.

*proof:* For each action pair  $\langle a_i, a_j \rangle$ , the relationships between  $a_i$  and  $a_j$  can be specified as whether or not, a proposition  $p$  in  $add_i, pre_i$ , or  $del_i$  is also in  $add_j, pre_j$ , or  $del_j$ . That is to say, there are nine kinds of relationships:  $\{p \in add_i \wedge p \in pre_j, p \in add_i \wedge p \in add_j, p \in add_i \wedge p \in del_j, p \in del_i \wedge p \in pre_j, p \in del_i \wedge p \in pre_j, p \in del_i \wedge p \in add_j, p \in del_i \wedge p \in del_j, p \in pre_i \wedge p \in pre_j, p \in pre_i \wedge p \in add_j, \text{ and } p \in pre_i \wedge p \in del_j\}$ . In another word, we only need to prove that  $\{p \in add_i \wedge p \in add_j, p \in del_i \wedge p \in pre_j, p \in del_i \wedge p \in del_j, \text{ and } p \in pre_i \wedge p \in add_j\}$  can be deduced by C1-C5, or they are contradictive with R1-R2.

First, if  $p \in add_i \wedge p \in add_j$  holds, then  $p \in preState_j$  holds. That is to say,  $p \in add_j \wedge p \in preState_j$  holds, which is contradictive with R1. Thus,  $p \in add_i \wedge p \in add_j$  is contradictive with R1.

Second, if  $p \in del_i \wedge p \in pre_j$  holds, then  $p \notin preState_j \wedge p \in pre_j$  holds. Since  $pre_j \subseteq preState_j$  holds,  $p \notin preState_j \Rightarrow p \notin pre_j$ , which implies  $p \notin pre_j \wedge p \in pre_j$ , i.e., contradiction is generated.

Third, if  $p \in del_i \wedge p \in del_j$  holds, then  $p \notin preState_j \wedge p \in del_j$  holds. And then  $p \notin preState_j \wedge p \in pre_j$  holds. Similar to the second one, contradiction will be generated.

Finally, from C3, we have  $p \in del_i \wedge p \in add_j$ . Then we have  $p \in pre_i \wedge p \in add_j$  by R2. On the other hand, if we have  $p \in pre_i \wedge p \in add_j$ , then we have  $p \in pre_i \wedge p \notin preState_j$  by R1, which implies  $p \in del_i$ . Thus,  $p \in del_i \wedge p \in add_j$  holds. That is to say,  $p \in pre_i \wedge p \in add_j$  is unnecessary, since C3 and R1-R2 have encoded the information it provides.

Briefly, by considering R1-R2, C1-C5 have encoded all the possible relationships between  $a_i$  and  $a_j$ .  $\square$

### 4.3 Assigning Weights

By steps 1-2, we have built a list of constraints. In this section, we present how to assign weights to constraints, which corresponds to steps 3-4 of Algorithm 1. Our basic idea is that (1) the weights of constraints built by step 1 should be high enough to ensure a planning problem being solved correctly; (2) the correct information included by the plan cases  $PC$  corresponds to the constraints frequently satisfied by  $PC$ , while the other information corresponds to the constraints infrequently satisfied. Thus, to explore the correct information, we calculate the frequency of constraints satisfied by  $PC$  as weights. Based on this idea, we give the algorithm of assigning weights to constraints in Algorithm 2.

In step 14 of Algorithm 2, each constraint in  $C_k$  is unified by substituting all the parameters of  $p, a_i$  and  $a_j$  with unified variables. We get the weights of constraints of C1-C5. Since we wish to solve a new planning problem correctly, we set weights

**Algorithm 2.** assigning weights to constraints**Input:** a planning domain  $\Sigma$ , a set of plan cases  $PC$ ;**Output:** the weights  $W_1, W_2, W_3, W_4, W_5$  for C1, C2, C3, C4, C5;

---

```

1:  $C_1 = C_2 = C_3 = C_4 = C_5 = \emptyset$  are sets of constraints of C1-C5 respectively;
2: for each plan case  $pc \in PC$  do
3:   for each two actions  $a_i$  and  $a_j$  in  $pc$  do
4:     if  $i < j$  and there is a proposition  $p$  that is not affected by actions between  $a_i$  and  $a_j$ 
       (can be asserted by executing the actions using  $\Sigma$ ) then
5:       for  $k = 1$  to  $5$  do
6:         if  $p, a_i$  and  $a_j$  form a constraint  $c$  that satisfies  $C_k$  then
7:           put  $c$  in  $C_k$ ;
8:         end if
9:       end for
10:     end if
11:   end for
12: end for
13: for  $k = 1$  to  $5$  do
14:   unify  $C_k$  into variable form;
15:   count the appearing number of each constraint in  $C_k$ ; the results are stored in a vector
        $W_k$ , viewed as weights;
16: end for
17: return  $W_1, W_2, W_3, W_4, W_5$ ;

```

---

(denoted as  $W_0$ ) of constraints (clauses, denoted as  $C$ ) generated in step 1 of Algorithm 1 as high as possible, by considering the effect of R1-R2 simultaneously. To do this, we first find the maximal value from  $W_1, W_2, \dots, W_5$ , which is denoted as  $w_{max}$ . Then, we set  $W_0$  by the following way:

$$\forall k > 0, W_0(k) = \beta w_{max}$$

where  $W_0(k)$  is a weight of  $k$ th constraint (clause) in  $C$ .  $\beta$  is a parameter to adjust the value of  $W_0(k)$ . By setting different value of  $\beta$ , the weights of constraints in  $C$  will be changed. As a result, the importance of the constraints in  $C$  will be changed correspondingly in the whole solving process of our algorithm MAXCBP. In our subsequent experiments, we will test different value of  $\beta$  to see its effect on the experiment result.

#### 4.4 Obtaining a Final Solution Plan

In steps 5-7 of Algorithm 1, we solve the weighted constraints of  $C$  and C1-C5 using a weighted MAX-SAT solver, the result of which is an assignment to all the axioms of constraints. With the assignment, we attain a plan by this way: first, we select all the axioms assigned with a *true* value; and then we convert all the selected axioms, which represent actions being executed or not, to a plan. Next, we will give a whole example for our algorithm MAXCBP in the following.

**Example:** For the planning problem in Table 1 the solving process of our algorithm MAXCBP is shown in Fig. 7. In this figure, the omitted parts denoted by “...” are the

```

---- Begin ----
Step 1: initial state and goal: (ontable A 1) ^ (clear A 1) ^ (holding B 1) ^ (on A B ∞) ^ (ontable B ∞)
      action: forall x, i. (pick-up x i) → (clear x i) ^ (ontable x i) ^ (handempty i)
      ...
      other constraints: forall x, x', i. (x ≠ x') → ¬(pick-up x i) ∨ ¬(pick-up x' i)
                       forall i < N, exists x. (pick-up x i)

Step 2: C1-C5 from plan cases 1 (likewise for plan case 2):
C1 add-pre constraints: (holding A 2) ∈ add(pick-up1) ^ (holding A 2) ∈ pre(stack2)
C2 add-del constraints: (holding A 2) ∈ add(pick-up1) ^ (holding A 2) ∈ del(stack2)
...

Step 3: assigning weights of C1-C5:
2 (holding A 2) ∈ add(pick-up1) ^ (holding A 2) ∈ pre(stack2)
2 (holding A 2) ∈ add(pick-up1) ^ (holding A 2) ∈ del(stack2)
...

Step 4: assigning weights of C:
β*2 (ontable A 1) ^ (clear A 1) ^ (holding B 1) ^ (on A B ∞) ^ (ontable B ∞)
β*2 ...

Step 5: solving weighted clauses from Step 3 and Step 4 using a weighted MAX-SAT solver, the
result is:
      (put-down B 1)   true
      (pick-up A 2)   true
      (stack A B 3)   true
      ...

Step 6: convert the result of Step 5 to a plan to the new planning problem:
      (put-down B) (pick-up A) (stack A B)

---- End ----

```

**Fig. 1.** An example of solving a new planning problem using MAXCBP

ones can be builded similarly by what are builded prior to them. In steps 3 and 4, the numbers “2” and “β\*2” are weights of C1-C5 and C respectively. In step 6, except the ones assigned to be true, there are other propositions assigned to be false or true (e.g., initial state and goal), which are not shown in the figure (denoted by “...”). From this example, we can see the detail steps about how to find a plan using a weight MAX-SAT solver.

## 5 Experiment Results

In this section, we evaluate our algorithm MAXCBP in the following three benchmark planning domains: *blocks*, *depots*<sup>2</sup> and *driverlog*<sup>2</sup>. We generated 150 plan cases from each domain. Furthermore, we generated 50 planning problems from each domain,

<sup>2</sup> <http://planning.cis.strath.ac.uk/competition/>

which will be solved by our algorithm MAXCBP. We evaluate MAXCBP by testing the running time and average length of all the plans according to different number of cases and different value of  $\beta$  which is a coefficient of  $w_{max}$ . Notice that we consider the efficiency and effectiveness of our algorithm MAXCBP by testing the running time and average length of plans respectively. We run our algorithm on the PC with CPU 2.26GHZ and memory 1GMB. We define the average length of all the plans as

$$A = \frac{\sum_{p \in P} lengthof(p)}{|P|}$$

where  $P$  is a set of plans and the procedure  $lengthof(p)$  returns the length of the plan  $p$ . In the following, we give the experimental results according to different number of cases and different values of  $\beta$ .

### 5.1 Different Number of Cases

In this experiment, we test the running time of our algorithm according to different number of plan cases being used in three different domains. The result is shown in Fig. 2 where the vertical line denotes the CPU time which is taken to solve all the 50 planning problems, and the horizontal line denotes the number of plan cases. In this figure, likewise for Fig. 4 the horizontal line signed with “without cases” shows the running result with the SATPLAN without exploiting any information of plan cases, and the curve signed with “with cases” shows the running result of our algorithm MAXCBP.

From Fig. 2 we find that the running time of our algorithm MAXCBP that exploits the information of plan cases is generally lower than the one without using any information. Furthermore, from the curves in Fig. 2 (a)-(c), we also find that the CPU time of our algorithm MAXCBP goes down when the number of plan cases goes up. That is because, the more the plan cases are given, the more the information can be used, that a plan can be found by MAXCBP more efficiently with the help of the information.

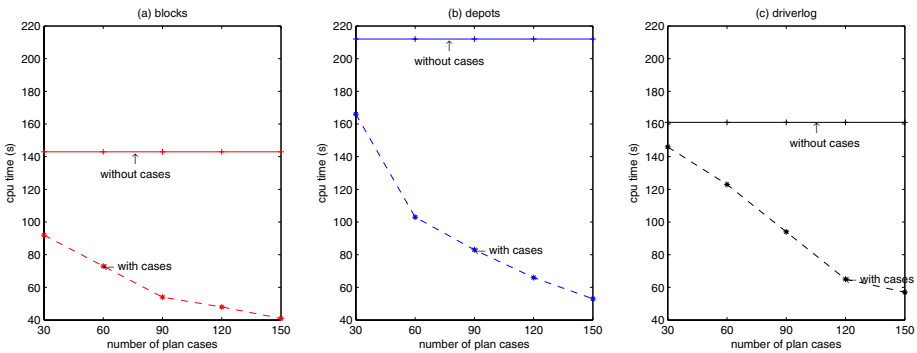
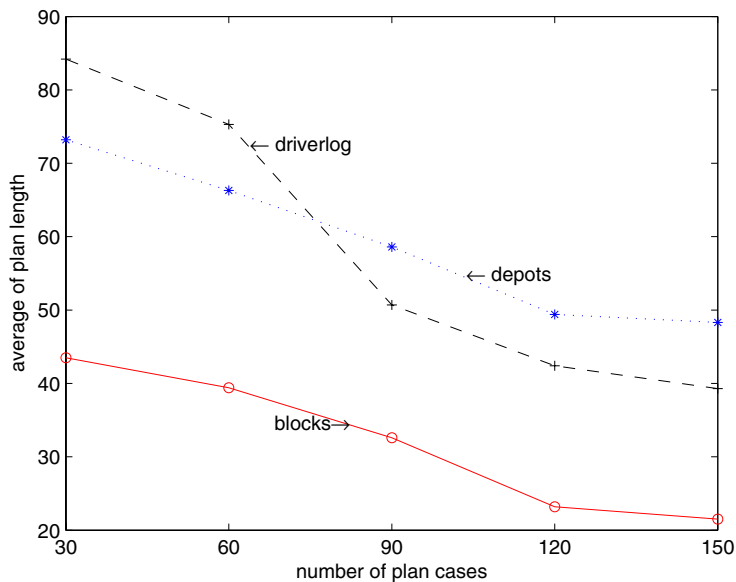


Fig. 2. The CPU time with respect to the number of plan cases

<sup>3</sup> <http://www.cs.rochester.edu/u/kautz/satplan/index.htm>





**Fig. 3.** The average of plan length with respect to the number of plan cases

To see the effect our algorithm MAXCBP introduces, we show the result of the average of plan length  $A$  with respect to the number of plan cases in Fig. 3. From this figure, we find that the curves are generally go down when the number of plan cases increases. That is because, the information of plan cases can help MAXCBP to find a *shorter* plan, rather than a *longer* one. Generally speaking, a *shorter* plan to a problem suggests that the problem is solved more efficiently (with fewer actions).

## 5.2 Different Values of $\beta$

In section 4.3, we assign the weights of the clauses  $C$  as  $W_0(k) = \beta w_{max}$ , where different  $\beta$  will result in different CPU time or plan length. The results can be seen from Fig. 4 and 5, where we fix the number of plan cases as 150 and test  $\beta$  with the values from 1 to 5.

For the CPU time in Fig. 4, we find that the CPU time is generally lower when using the information of plan cases than without it, by comparing the curves denoted with “with cases” for planning using the case base, and the horizontal lines denoted with “without cases” to denote planning from first principles without using the case base, from Fig. 4(a)-(c). After testing the weight factor  $\beta$ , we find that the CPU time generally increases with the value of  $\beta$ . That is because, when  $\beta$  increases, the information that plan cases can provide is reduced, which means MAXCBP will take more CPU time to find a plan when using less information from the plan cases.

For the plan length in Fig. 5, we find that the average length of plans for all the 50 problems is generally going up when the value of  $\beta$  increases. Similar to Fig. 4, when

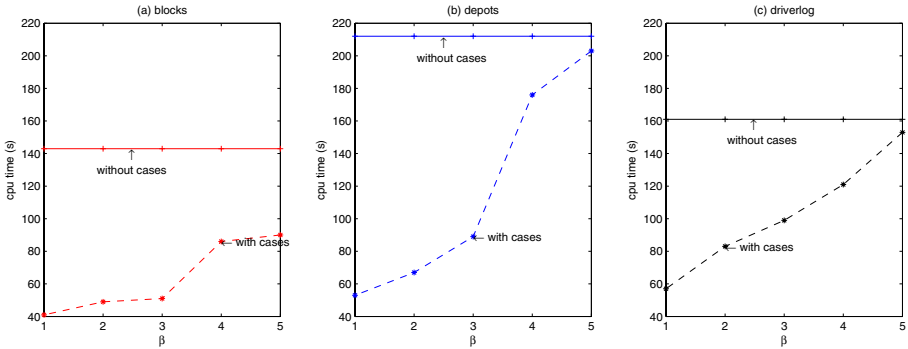


Fig. 4. The CPU time with respect to different values of  $\beta$

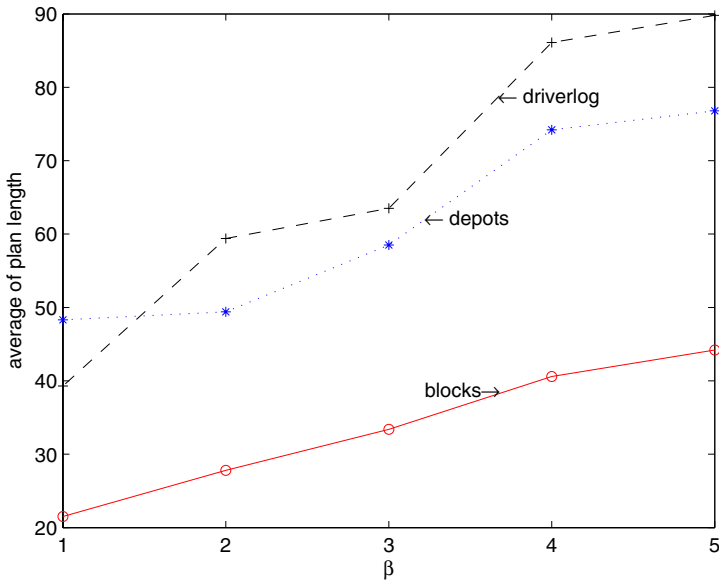


Fig. 5. The average length of plans with respect to different values of  $\beta$

$\beta$  increases, the effect of plan cases on helping finding a shorter plan is weakened, that the plans being found by MAXCBP will be longer.

From Fig. 245, we conclude that, our algorithm MAXCBP, which is to maximally exploit the information of plan cases, will help improve the efficiency and effectiveness of finding a plan to a new planning problem.

## 6 Conclusion

In this paper, we presented a novel approach for case-based planning called MAXCBP. Our algorithm makes maximal use of the cases in the case base to find a solution by

maximally exploiting the information of plan cases via using a weighted MAX-SAT solver. Our system can take a piece of the useful plan knowledge in the form of constraints even when the entire plan may not be useful for solving a new problem. Our empirical tests show that our method is both efficient and effective in solving a new planning problem. In real world applications, attaining a set of plan cases by hand is difficult and time-consuming. Thus, in our future work, we will consider the situation that plan cases are observed automatically by machine such as sensors. In this situation, plan cases will contain noise, which makes our task more difficult. Thus, one of our future works is to extend the framework by considering more noisy cases.

## Acknowledgment

We thank the support of Hong Kong CERG Grant HKUST 621307, NEC China Lab.

## References

1. Kuchibatla, V., Muñoz-Ávila, H.: An Analysis of Transformational Analogy: General Framework and Complexity. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS, vol. 4106, pp. 458–473. Springer, Heidelberg (2006)
2. Kautz, H., Selman, B.: Planning as Satisfiability. In: ECAI (1992)
3. Borchers, B., Furman, J.: A Two-Phase Exact Algorithm for MAX-SAT and Weighted MAX-SAT Problems. *Journal of Combinatorial Optimization* 2(4), 299–306 (1998)
4. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 463–502 (1969)
5. Munoz-Avila, H., Cox, M.T.: Case-Based Plan Adaptation: An Analysis and Review. *IEEE Intelligent Systems* (2007)
6. Blum, A.L., Furst, M.L.: Fast planning through planning graph analysis. *Artificial Intelligence* (90), 1636–1642 (1997)
7. Hammond, K.J.: *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, San Diego (1989)
8. Sugandh, N., Ontanon, S., Ram, A.: On-Line Case-Based Plan Adaptation for Real-Time Strategy Games, pp. 702–707. AAI, Menlo Park (2008)
9. Kuchibatla, V., Muñoz-Ávila, H.: An Analysis on Transformational Analogy: General Framework and Complexity. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS, vol. 4106, pp. 458–473. Springer, Heidelberg (2006)
10. Bajo, J., Corchado, J.M., Rodriguez, S.: Intelligent Guidance and Suggestions Using Case-Based Planning. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 389–403. Springer, Heidelberg (2007)
11. de la Rosa, T., García Olaya, A., Borrajo, D.: Using cases utility for heuristic planning improvement. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 137–148. Springer, Heidelberg (2007)
12. Petrik, M., Zilberstein, S.: Learning Heuristic Functions Through Approximate Linear Programming. In: ICAPS (2008)
13. de la Rosa, T., Jimenez, S., Borrajo, D.: Learning Relational Decision Trees for Guiding Heuristic Planning. In: ICAPS (2008)
14. Yoon, S., Fern, A., Givan, R.: Learning Control Knowledge For Forward Search Planning. *JMLR* 9(APR), 683–718 (2008)

15. Fikes, R., Nilsson, N.J.: Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 189–208 (1971)
16. Yang, Q.: *Intelligent Planning: A Decomposition and Abstraction Based Approach*. Springer, Berlin (1997)
17. Chapman, D.: Planning for Conjunctive Goals. *Artificial Intelligence* 32, 333–377 (1987)
18. Wilkins, D.E.: Recovering from Execution Errors in SIPE. *Computational Intelligence* 1, 33–45 (1985)
19. Selman, B., Levesque, H., Mitchell, D.: Hard and Easy Distributions of SAT Problems. In: *Proc. of the 10th National Conference on Artificial Intelligence*, San Jose, CA, July 1992, pp. 440–446. AAAI Press/MIT Press (1992)
20. Carbonell, J.G.: Learning by analogy: formulating and generalizing plans from past experience. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, California (1983)

# A Value Supplementation Method for Case Bases with Incomplete Information

Kerstin Bach, Meike Reichle, and Klaus-Dieter Althoff

Intelligent Information Systems Lab  
University of Hildesheim  
Marienburger Platz 22, 31141 Hildesheim, Germany  
{bach,reichle,althoff}@iis.uni-hildesheim.de

**Abstract.** In this paper we present a method for supplementing incomplete cases with information from other cases within a case base. The acquisition of complete and correct cases is a time-consuming task, but nevertheless crucial for the quality and acceptance of a case-based reasoning system. The method introduced in this paper uses association rules to identify relations between attributes and, based on the discovered relations we are able to supplement values in order to complete cases. We argue that using these related attributes when retrieving supplementation candidates will yield better results than simply picking the case with the highest global similarity. The evaluation of the method is carried out using four different publicly available case bases.

## 1 Introduction

Incomplete information in cases is a problem often encountered in various areas of Case-Based Reasoning (CBR). For instance, Bogaerts and Leake [1] discuss how to assess the similarity of incomplete problem descriptions in Conversational CBR applications, while Selvamani and Khemani [2] discuss how missing information can be completed using decision tree induction. Further on when cases are collected from WWW sources, like blogs, websites or web communities as in [3], completing cases or dealing with incomplete information is one of the major challenges.

Missing attributes can happen for a number of reasons. For instance, considering products as cases as within our example case bases, attributes can be empty either because the attribute's value is unknown or because a certain attribute doesn't apply to a certain case/product. Obviously only the first group should be substituted, so a substitution strategy should ideally also include a set of rules or constraints that controls when an empty attribute is substituted and when a substitution is not applicable. Such constraints could for example be based on other attribute's values ("If a PC is a Desktop do not substitute the battery attribute") or a similarity-based comparison with other products and their missing values.

In this paper we present a method for supplementing incomplete cases in Structured CBR (SCBR) applications [4] using only the case base itself, the

knowledge already included in the cases and the similarity model. In SCBR a case can be represented as attribute-value tables, in an object-oriented manner, trees or graphs as well as in predicate logic [5]. We focus on the representation in attribute-value tables, because it is one of the most common kind of case representation in CBR. Cases in SCBR are represented by a predefined set of attributes and the range of the attributes' values (mostly nominal and numerical) is given in a vocabulary [6].

This paper picks up on previous experiments on the subject, raises them to a more general level, evaluates the results of the method and identifies constraints on its applicability.

The aim of this paper is to demonstrate that – given the necessity to supplement cases – our method leads to more accurate supplementations than doing a standard CBR retrieval on the case to be supplemented and simply supplementing it with the attribute values from the most similar case. Our method provides a result set which is optimized towards retrieving the best fitting supplementation candidates, even if they are not actually the most similar cases.

The work in this paper is structured as follows: Section 2 gives a short overview on the preliminary work already carried out on the subject and the first practical results achieved within our project docQuery [7]. Section 3 presents and evaluates the method in a more generalized way: in subsection 3.1 we describe the experiments used to evaluate the general applicability of our method, followed by the presentation of the results of the individual experiments in subsection 3.2 and the interpretation of the results as well as a concluding estimation of its general applicability in section 3.3. Section 4 presents related work on comparable topics. The paper concludes with a brief summary and an outlook on future work in section 5.

## 2 Preliminary Work

We initially presented this approach in [8] as an improvement to adaptation in case bases consisting of a complete set of cases but with individual cases suffering from incomplete information. In that first scenario we dealt with a geographic case base, which was used in the context of travel medicine. This case base included cases for all known countries but with a heavily varying information quality, i.e. some attributes were always present, such as the vaccinations that are obligatory in order to enter a country, others were often empty. Since the application required a complete case for its next steps to work out we had to develop a method for filling those attributes with values. A closer study of the case format revealed that there are certain attributes that are related with regard to their content, in that case for instance the necessary vaccinations and the general list of infection risks. The content of the vaccinations attributes suggested, at least partly, the contents of the general infection risk, and thus lent itself to be used in order to derive a sensible value for that attribute.

Assuming that similar vaccinations suggest similar general infection risks, we then developed a 2-step retrieval method in order to find the optimal case from

which to take over the necessary values. Applied to the given example the method was carried out as follows:

1. Select the desired country from the case base. Since we identify the countries by name, which is a unique attribute, this can be done using similarity based retrieval as well as a simple selection.
2. If the country's general infection risks are not indicated, extract the content of the vaccinations attribute.
3. Send a new query, this time using the country's name and vaccinations as query input.
4. Take the result set, remove the best hit (which will again be the country in question), randomly pick one of the remaining countries with the highest similarity.
5. Extract the randomly picked country's general infection risks and use them to supplement the original country's information. If the picked country also has an empty general infection risks attribute pick another case of the same similarity.

In order to evaluate the quality of the resulting supplementations we manually prepared a test case base (countries of South East Asia) with complete information and then subsequently took every country, emptied its general infection risks and restored them using once the 2-step retrieval method and once using only a geographic taxonomy as the similarity measure in order to pick out a supplementation candidate.

Using the 2-step retrieval method we were able to significantly reduce the number of supplementation candidates in 90% of the test. Evaluating the quality of the supplementations done with the respective remaining candidates we found that in a total of 90 supplementations using the taxonomy based retrieval 62% of the supplemented cases contained all of the expected infection risks. Using the 2-step retrieval method on the same cases amounted in 76% of the supplemented cases containing all expected infection risks. Although both retrieval variants also returned false positives in most of the tests, the solutions of the 2-step retrieval method were generally more reliable, especially with respect to false negatives, which were the more serious problem in this particular application scenario.

Our application scenario and its underlying architecture SEASALT [3] uses modularized knowledge bases. For the combination of information retrieved from these knowledge bases we do a subsequent retrieval and the more information one solution contains, the more possibilities the algorithm has for further retrieval steps. If we would only have a single retrieval step, a more precise similarity measure including specialised domain knowledge would probably work as well.

### 3 Generalization

Phrased more generally, according to the 4-R model [9] our supplementation steps in after the *retrieve* step, but before a potential *reuse* step. Thus in our point of view we replace the general *retrieve* step, with two steps: first we retrieve

the desired case as usual, but then we check it for completeness, and, if the values of a related attribute are missing, we do a second retrieval in order to get a supplementation candidate case. We then use its values to supplement the desired case and finally pass it on to *reuse*, *revise*, and *retain*.

In order to see if the method is generally suited for supplementing incomplete cases we identified the following research hypotheses:

1. *There are pairs of attributes that are related with respect to their content, i.e. the value of one attribute determines – with a certain confidence – the value of the other attribute. If a case format includes such relations between attributes they can . . .*
  - (a) *be identified automatically and*
  - (b) *be used to supplement missing values of related attributes.*
2. *The supplementation candidates retrieved using only related attributes will be the most fitting, i.e. the results of the supplementation will be better than the results when using a global similarity measure to retrieve the supplementation candidates.*

### 3.1 Experiments towards an Extensions of the Initial Scenario

In order to be as representative as possible we used publicly available test case bases from UCD, namely the PC and Whiskey case bases [10], the camera case base [11] and AI-CBR's travel case base<sup>1</sup>. These case bases each provide a set of cases as well as a case format and the associated similarity measures. We used the models and measures as indicated.

The camera case base consists of 210 case described with four nominal and six numerical attributes. The numerical attributes have a predefined range depending on their values. The global similarity measure is calculated using each attribute with the same weight. The whiskey case base consists of 553 cases and each case is represented by ten attributes, five nominal and five numerical attributes. The global similarity measure is calculated using each attribute with the same weight. The PC case base contains 120 cases which are represented by eight attributes and the global similarity measure is calculated using each attribute with the same weight. Three attributes are nominal and five are numerical with a defined range of possible values. The travel case base contains 1024 cases and the case representation consists of six nominal and three numerical attributes. The global similarity measure is calculated using each attribute with the same weight.

As a first step we calculated the related attributes. For this purpose we used a simple 1R algorithm [12] for finding association rules within the cases of one case base. Then we compared for each attribute combination the respective rules with a confidence  $\geq 67\%$  against the total number of all value combinations for those attributes within the case base, resulting in a final correlation score.

---

<sup>1</sup> We gathered the XML files describing the case bases from the Case-Based Reasoning Wiki at [http://cbrwiki.fdi.ucm.es/wiki/index.php/Case\\_Bases](http://cbrwiki.fdi.ucm.es/wiki/index.php/Case_Bases)



$$\text{c-score}_{(A1,A2)} = \frac{\# \text{ of rules from A1 to A2 with a confidence } \geq 67\%}{\# \text{ of combinations from A1 to A2 within the case base}} * 100$$

For example considering the Whiskey case base there are 193 different combinations of values of the attributes *Proof* and *Finish*. 77 of these combinations could serve as an association rule with a confidence  $\geq 67\%$ . Thus the c-score for *Proof*  $\rightarrow$  *Finish* amounts to 39.9.

We then iterated over each case base and, for each case, we tried to supplement the identified attributes using the respective related attributes. We did this twice for each case, once using the whole case for the second query, once using only the identified related attribute. In order to simulate an incomplete case base we each time randomly removed three attribute values from the complete cases, but never the related attribute, so the supplementation in the first test was based on the retrieval results using all remaining attributes. The second test supplementation was based on a query using only the related attributes.

To illustrate this with an example let's consider the camera case base. In this case base one of the highest rating attribute pairs was *Weight*  $\rightarrow$  *Format*, we will thus try to supplement the *Format* attribute. We consider the case listed in table 1 and try to supplement the value of *Format*. In the first test we simply use the whole case in order to find the most similar camera and use its *Format* for supplementation. The result set naturally lists the original case first, then, with the second best amount of similarity, there are two supplementation candidate cases. One of these cases has the desired value "SLR" and the other one "Compact", so there would have been a 50% chance of a correct supplementation. In

**Table 1.** Example case from the camera case base. *Format* is the attribute to be supplemented, *Manufacturer*, *Model* and *Storage Included* have been randomly removed.

Attribute	Value
ID	Case140
Manufacturer	- removed -
Model	- removed -
Price (\$)	900
Format	- missing -
Resolution (M Pixels)	1.92
Optical Zoom (X)	3.2
Digital Zoom (X)	2
Weight (grams)	630
Storage Type	Compact Flash
Storage Included (MB)	- removed -

the second test we only use the related attribute *Weight* for retrieval. The result is computed analogously. This time the chance of a correct supplementation would have been 71.42%, because 10 out of 14 supplementation candidate cases suggest the correct value.

We did this once for each case in each of the four case bases using the following attribute pairs<sup>2</sup>:

- Camera
  - *Camera.Weight* → *Camera.Format*
  - *Camera.Weight* → *Camera.OpticalZoom*
  - *Camera.Weight* → *Camera.StorageType*
  - Excluded attributes: *CaseId* (unique, avg. frequency 1), *Model* (avg. frequency 1.005), *Price* (avg. frequency 1.2)
- PC
  - *PC.Monitor* → *PC.Type*
  - *PC.DriveCapacity* → *PC.Type*
  - *PC.ProcessorSpeed* → *PC.ProcessorType*
  - Excluded attributes: *Price* (avg. frequency 1.5)
- Travel
  - *Travel.Hotel* → *Travel.Accommodation*
  - *Travel.Hotel* → *Travel.Region*
  - *Travel.Hotel* → *Travel.Transportation*
  - *Travel.Region* → *Travel.Transportation*
  - Excluded attributes: *Price* (avg. frequency 1.6)
- Whiskey
  - *Whiskey.Proof* → *Whiskey.Finish*
  - *Whiskey.Proof* → *Whiskey.Availability*
  - *Whiskey.Proof* → *Whiskey.Sweetness*
  - *Whiskey.Proof* → *Whiskey.Peatiness*
  - Excluded attributes: none

### 3.2 Results

Considering the supplementation results it can be noted that our first research hypothesis (*There are pairs of attributes that are related with respect to their content, i.e. the value of one attribute determines – with a certain confidence – the value of the other attribute. If a case format includes such relations between attributes they can (a) be identified automatically and (b) be used to supplement missing values of related attributes.*) is confirmed by the results of our experiments. We were able to reliably detect meaningful correlations between attributes and with very few exceptions the attribute pairs with the highest correlation score were also the ones with the best supplementation results. Even

<sup>2</sup> Attribute pairs including unique or almost unique attributes (i.e. attributes with values with an average frequency near 1) were manually excluded.

when supplementation using all available attributes performed better overall, the difference between the results is smaller the higher the correlation score is.

Our second research hypothesis (*"The supplementation candidates retrieved using only related attributes will be the most fitting, i.e. the results of the supplementation will be better than the results when using other retrieval methods to retrieve the supplementation candidates."*) only held for some of the case bases.

In detail the results were as follows: In the Camera case base (see results in table 2) supplementation based on a retrieval using all available attributes outperformed supplementation based on a search using only the related attribute in all of the tests. In the Whiskey case base (see results in table 3) supplementation

**Table 2.** Successful supplementations in the camera case base: based on all available attributes vs. using only the related attribute

Att Pair	Correlation Score	Correct supps all atts [%]	Correct supps only rel att [%]	Improvement [%]
<i>Weight</i> → <i>Format</i>	68.52	76.00	63.00	-13.00
<i>Weight</i> → <i>OpticalZoom</i>	56.10	63.00	47.00	-16.00
<i>Weight</i> → <i>StorageType</i>	50.00	72.00	30.00	-42.00
<b>Camera Case Base</b>				

**Table 3.** Successful supplementations in the whiskey case base: based on all available attributes vs. using only the related attribute

Att Pair	Correlation Score	Correct supps all atts [%]	Correct supps only rel att [%]	Improvement [%]
<i>Proof</i> → <i>Finish</i>	39.90	47.00	66.66	19.66
<i>Proof</i> → <i>Availability</i>	38.01	18.90	18.75	-0.15
<i>Proof</i> → <i>Sweetness</i>	32.96	26.10	10.40	-15.70
<i>Proof</i> → <i>Peatiness</i>	32.60	22.70	8.30	14.40
<b>Whiskey Case Base</b>				

based on a retrieval using all available attributes was outperformed by supplementation based on a search using only the related attribute in 25% of the tests. In the PC case base (see results in table 4) supplementation based on a retrieval using all available attributes was outperformed by supplementation based on a search using only the related attribute in 66% of the tests. The outperformed attribute pair was also the one with the lowest correlation score. In the Travel case base (see results in table 5) supplementation based on a retrieval using all available attributes was outperformed by supplementation based on a search using only the related attribute in 100% of the tests. The result tables each indicate the used attribute pair,

**Table 4.** Successful supplementations in the PC case base: based on all available attributes vs. using only the related attribute

Att Pair	Correlation Score	Correct supps all atts [%]	Correct supps only rel att [%]	Improvement [%]
<i>Monitor</i> → <i>Type</i>	46.67	51.00	65.00	14.00
<i>DriveCapacity</i> → <i>Type</i>	45.00	45.30	46.70	1.40
<i>ProcSpeed</i> → <i>ProcType</i>	37.50	76.00	56.00	-20.00
<b>PC Case Base</b>				

**Table 5.** Successful supplementations in the travel case base: based on all available attributes vs. using only the related attribute

Att Pair	Correlation Score	Correct supps all atts [%]	Correct supps only rel att [%]	Improvement [%]
<i>Hotel</i> → <i>Accommodation</i>	99.44	64.37	99.38	36.01
<i>Hotel</i> → <i>Region</i>	98.04	41.68	98.57	56.89
<i>Hotel</i> → <i>Transportation</i>	90.21	88.22	92.76	4.54
<i>Region</i> → <i>Transportation</i>	77.33	89.46	93.53	10.07
<b>Travel Case Base</b>				

their respective correlation score and the percentage of correct supplementations in both tests as well as the difference between both tests (Improvement).

### 3.3 Evaluation of the Experiments and Their Results

Summarizing the results of all four case bases we think that our supplementation method shows promise also on a more general level. The very different nature of the case bases used in these experiments allows us to draw first conclusions with regard to the general applicability of our method. We are very satisfied with the results of our identification of related attribute couples. Although it uses a very simple algorithm and is overall implemented in a rather pragmatic way it achieves very good results. The method also seems to perform equally well on case bases with more and fewer numerical attributes. An additional benefit of detecting such related attributes is the possibility to use this knowledge in order to improve the system's similarity model. The determining attributes obviously possess a high information value. On the other hand the attributes that can be deduced from them are not necessarily redundant but have more of a supporting role. This knowledge can for instance be reflected in assigning a higher weight to the determining attributes and a lower one to the supporting attributes. By recalculating the correlation scores on a regular basis and automatically adapting the attribute weights accordingly the correlation score provides an easy way of improving the similarity model along with the case base's competence.

Regarding the supplementation results, the Travel case base's results are obviously best, since the identified attribute pairs are very closely related, which is also reflected in the very high correlation scores. However there are also case bases in which a retrieval with all available attributes performs better than our method, the most prominent example being the Camera case base. We assume that these bad results of our method are caused by weak attribute pairs, i.e. an inexact computation of correlation scores. The Camera case base is comparatively small (210 cases) but has a rather high number of attributes (10), many of which again have a large range of possible values. This means that the majority of possible value combinations is not covered in the case base and that the amount of covered combinations might even not be representative. As presented by MacDonald et. al. [13] a high number of attributes and possible values also requires a high number of cases. Otherwise the minimum similarity threshold has to be specified so low that result quality is no longer acceptable. We assume that the same holds for attribute correlations.

Concerning the applicability of our approach there are of course certain limits, mostly with respect to the application domain. Most of all there have to be attributes which are related with regard to their content, but it should also be kept in mind that any supplementation method comes down to more or less educated guessing and thus should not be used in domains that are safety critical or require high precision data. Also, as mentioned in the introductory notes, not all missing values require supplementation. Attributes that don't necessarily apply in any case should possess a null value that indicates a deliberately empty value. Also attributes with a high similarity weight could be treated more carefully

(e.g. by requiring a higher confidence value when doing supplementations) in order to avoid too much of an effect on the retrieval results.

Finally it would be advisable to reflect the fact that a case has been supplemented in the case's description, thus allowing the ranking mechanism to favour more complete and thus more reliable cases and also creating a higher transparency towards the user.

## 4 Related Work

Several researchers have presented works on topics related to the work presented in this paper. Data Mining or Knowledge Discovery techniques have already been combined with CBR in the past. O'Sullivan et. al. [14] used data mining algorithms to maintain similarity knowledge in order to improve case-based collaborative filtering recommendations. Díaz-Agudo et. al. [15] used the Formal Concept Analysis (FCA) as an inductive technique to extract domain specific knowledge from cases. They used FCA in knowledge intensive applications to enrich domain ontologies or change the organization of case bases.

Dubois et al. [16] also consider similarities between a case's attributes and combine that knowledge with fuzzy logic in order to deal (among other things) with incomplete cases.

The relations between attributes have also been investigated by Tawfik and Kasrin [17] who represent them using dependency graphs which are then sectioned using either d-separation [18] or multiply sectioned Bayesian networks [19]. Tawfik and Kasrin use the resulting subgraphs/-cases to generate completely new cases for the purpose of increasing case base coverage. By using the subgraphs they aim to detect dependencies between attributes and thus prevent intra-case inconsistencies [20] when generating new cases.

Redmond [21] introduced an approach for combining information from different cases. They define a case as a set of information pieces, like snippets in [22], consisting of an attribute-value-pair. Each snippet is assigned to a particular goal and holds information on how to pursue this goal. Since the reference application originates in CBR-diagnosis, the snippets also contain information (links) that preserve the structure of the diagnosis. Further on, they use a case-based reasoning process to retrieve single snippets and based on the predefined links they put together a problem's solution. The snippet information is highly dependent on the domain and has to be modeled by hand. The approach presented in our paper focuses on a more general method that also uses information from different cases by employing knowledge contained in the different knowledge containers [23], without explicitly modeling additional case information. However, both approaches have in common that the reasoning processes are used to supplement incomplete information in order to find better solutions.

The approach of doing several, subsequent retrieval steps instead of only one can also be found in the works of several authors. Weibelzahl [24,25] present an approach based on two different case bases. On the one hand they use a specific case base to create an enriched query that uses the given information more effectively and on the other hand they do regular CBR. They evaluate the approach

in a system on holiday recommendation consisting of two case bases with different knowledge models. The first case base, called customer case base, holds information on the customers' needs and desires which are mapped to attributes describing products provided in the second case base. In the first step the query containing the user's expectations on their vacation is analysed in order to fill relevant attributes creating a request which can be sent to the product case base. The second request contains especially those product attributes which the user would not request on their own, but which help to find an appropriate solution in the product case base. In comparison to our approach, we use the case base's knowledge model to enhance the query aiming at a more differentiated result while Weibelzahl points out that users cannot exactly describe their desires by framing a request. The incremental approach kind of matches the users statement to correct attribute-value pairs.

A similar approach is presented by Cunningham et al. in [26,27]. They introduce the Incremental CBR (I-CBR) mechanism for diagnosis. The I-CBR approach separates information in "free" and "expensive" features and starts the first retrieval steps based on the free features before the user is asked to give information about expensive features to narrow the set of cases. In comparison with their approach we have a different point of view. The method presented in our paper is able to indicate attributes that are determining for the retrieval and those which can be derived from the knowledge within the case base. In contrast to Cunningham's method, our approach does not classify "free" and "expensive" attributes; instead we are able to supplement missing information and thus do not require "expensive" information at all.

Another approach on how I-CBR can influence the result sets has been presented in [28], but in comparison to our approach Jurisica et. al. did not receive additional information from existing cases, they used query series and user interaction instead.

## 5 Conclusion and Outlook

Case acquisition is an extensive and time-consuming task, and often has to deal with incomplete information, resulting in incomplete cases. Supplementing such incomplete cases with information adapted from other cases is a relatively easy way to improve a case base. However, when carrying out such a supplementation the choice of which cases to supplement from is of paramount importance, since a wrong supplementation may actually worsen a case's information quality or even make it inconsistent. Association rules are a handy tool to identify attributes that are related with regard to their content, a knowledge which can be used well when choosing the optimal candidate for a supplementation.

In this paper we presented a method for supplementing incomplete cases using attributes from other cases that makes use of association rules and similarity based retrieval in order to pick an optimal supplementation candidate. On the one hand our method produces better supplementation candidates than using a CBR system's standard case retrieval and global similarity measure, since

it focuses on the related attributes. On the other hand, it could neither be replaced by a simple rule-based approach since it makes use of the CBR system's underlying similarity model and thus, if no valid supplementation candidate can be found, it will at least come up with a most similar value instead of none.

After having tested the method in one of our projects already, we now evaluate it using publicly available test case bases. The results of these evaluations are promising in so far as that our research hypotheses hold and the method performs well in most scenarios. However not all results are good, so there is still room for further research and improvement.

This further research will for instance concern the question under which circumstances the identification of related attributes works best, i.e. if there is a minimum number of cases and/or attribute combinations necessary in order for our method to yield satisfactory results. We will also do a few experiments on other association rule learning algorithms in order to find out how far the relative simplicity of the 1R algorithm influences the overall result quality. Apart from other association rule learning algorithms we will also evaluate our method against other substitution methods, e.g. not randomly picking the substitution value from the substitution candidates but using the most frequent value or a mathematic mean.

We are also interested in trying our method on other case bases in order to gain even more insight in the conditions of its general applicability and its behavior under different conditions. It would be especially interesting to evaluate our method with a more realistic test case base, that is a case base where we don't have to randomly delete values but already have missing values that derive from an real life application and are thus not as uniformly distributed. Alternatively, if such a test case base is not available, we will do some more experiments with varying amounts of removed values.

Another aspect that will receive greater attention in our future work are the possibilities to integrate the correlation score in automated improvement of the system's similarity model, as already sketched out in section 3.3 and possibly other areas of CBR research such as maintenance and adaptation. Finally a topic that might become more relevant in future experiments is the performance of our method and whether/how it can be improved with regard to computation time.

## References

1. Bogaerts, S., Leake, D.B.: Facilitating CBR for incompletely-described cases: Distance metrics for partial problem descriptions. In: Funk, P., González-Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 62–76. Springer, Heidelberg (2004)
2. Selvamani, R.B., Khemani, D.: Decision tree induction with CBR. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 786–791. Springer, Heidelberg (2005)
3. Bach, K., Reichle, M., Althoff, K.D.: A domain independent system architecture for sharing experience. In: Proceedings of LWA 2007, Workshop Wissens- und Erfahrungsmanagement, September 2007, pp. 296–303 (2007)



4. Börner, K., Pippig, E., Tammer, E.C., Coulon, C.H.: Structural similarity and adaptation. In: Smith, I.F.C., Faltings, B. (eds.) EWCBR 1996. LNCS, vol. 1168, pp. 58–75. Springer, Heidelberg (1996)
5. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. LNCS, vol. 2432. Springer, Heidelberg (2002)
6. Bergmann, R., Althoff, K.D., Minor, M., Reichle, M., Bach, K.: Case-based reasoning - introduction and recent developments. *Künstliche Intelligenz: Special Issue on Case-Based Reasoning* 23(1), 5–11 (2009)
7. Bach, K.: docquery - a medical information system for travellers. Internal project report (September 2007)
8. Reichle, M., Bach, K.: Improving result adaptation through 2-step retrieval. In: Nalepa, G.J., Baumeister, J. (eds.) Proceedings of the 4th Workshop on Knowledge Engineering and Software Engineering (KESE 2008) at the 31st German Conference on Artificial Intelligence (KI 2008), September 2008, pp. 73–84 (2008)
9. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* 1(7) (March 1994)
10. McGinty, L., Smyth, B.: Adaptive selection: An analysis of critiquing and preference-based feedback in conversational recommender systems. *Int. J. Electron. Commerce* 11(2) (06-7), 35–57
11. McCarthy, K., Reilly, J., McGinty, L., Smyth, B.: Experiments in dynamic critiquing. In: *IUI 2005: Proceedings of the 10th international conference on Intelligent user interfaces*, pp. 175–182. ACM, New York (2005)
12. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 63–91 (1993)
13. MacDonald, C., Weber, R., Richter, M.M.: Case base properties: A first step. In: Schaaf, M. (ed.) *ECCBR Workshops*, pp. 159–170 (2008)
14. O'Sullivan, D., Wilson, D.C., Smyth, B.: Improving case-based recommendation: A collaborative filtering approach. In: *Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416*, pp. 278–291. Springer, Heidelberg (2002)
15. Díaz-Agudo, B., González-Calero, P.A.: A declarative similarity framework for knowledge intensive CBR. In: *Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS, vol. 2080*, pp. 158–172. Springer, Heidelberg (2001)
16. Dubois, D., Esteva, F., Garcia, P., Godo, L., Mántaras, R.L.d., Prade, H.: Case-based reasoning: A fuzzy approach. In: *L. Ralescu, A. (ed.) IJCAI-WS 1997. LNCS, vol. 1566*, pp. 79–90. Springer, Heidelberg (1999)
17. Tawfik, A.Y., Kasrin, N.: Integrating causal knowledge in case-based retrieval: Causal decomposition of cases. In: *Petridis, M., Wiratunga, N. (eds.) Proceedings of the Thirteenth UK Workshop on Case Based Reasoning (UKCBR 2008)*. University of Cambridge, UK (2008)
18. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
19. Xiang, Y., Poole, D., Beddoes, M.P.: Multiply sectioned bayesian networks and junction forests for large knowledge-based systems. *Computational Intelligence* 9, 171–220 (1993)
20. Racine, K., Yang, Q.: On the consistency management of large case bases: the case for validation. In: *Proceedings of the AAAI 1996 Workshop on Knowledge Base Validation, American Association for Artificial Intelligence (AAAI)*, pp. 84–90 (1996)
21. Redmond, M.: Distributed cases for case-based reasoning: Facilitating use of multiple cases. In: *AAAI*, pp. 304–309 (1990)

22. Kolodner, J.L.: Retrieving events from a case memory: A parallel implementation. In: Proc. of a Workshop on Case-Based Reasoning, Holiday Inn, Clearwater Beach, FL, pp. 233–249 (1988)
23. Richter, M.M.: Introduction. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) *Case-Based Reasoning Technology*. LNCS (LNAI), vol. 1400, p. 1. Springer, Heidelberg (1998)
24. Weibelzahl, S.: Conception, implementation, and evaluation of a case based learning system for sales support in the internet. Master's thesis, Universität Trier (1999)
25. Weibelzahl, S., Weber, G.: Benutzermodellierung von Kundenwünschen durch Fall-basiertes Schliessen. In: Jörding, T. (ed.) *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, ABIS 1999, Magdeburg, pp. 295–300 (1999)
26. Cunningham, P., Bonzano, A., Smyth, B.: An incremental case retrieval mechanism for diagnosis (1995)
27. Cunningham, P., Smyth, B., Bonzano, A.: An incremental retrieval mechanism for casebased electronic fault diagnosis (1998)
28. Jurisica, I., Glasgow, J., Mylopoulos, J.: Incremental iterative retrieval and browsing for efficient conversational CBR systems. *Applied Intelligence* 12(3), 251–268 (2000)

# Efficiently Implementing Episodic Memory

Nate Derbinsky and John E. Laird

University of Michigan  
2260 Hayward Street  
Ann Arbor, MI 48109-2121  
{nlderbin, laird}@umich.edu

**Abstract.** Endowing an intelligent agent with an episodic memory affords it a multitude of cognitive capabilities. However, providing efficient storage and retrieval in a task-independent episodic memory presents considerable theoretical and practical challenges. We characterize the computational issues bounding an episodic memory. We explore whether even with intractable asymptotic growth, it is possible to develop efficient algorithms and data structures for episodic memory systems that are practical for real-world tasks. We present and evaluate formal and empirical results using Soar-EpMem: a task-independent integration of episodic memory with Soar 9, providing a baseline for graph-based, task-independent episodic memory systems.

## 1 Introduction

Episodic memory, as first described by Tulving [23], is a long-term, contextualized store of specific events. Episodic memory, what an individual “remembers,” is contrasted with semantic memory, a long-term store of isolated, de-contextualized facts that an individual “knows.” As an example, a memory of viewing artwork during one’s last vacation would be episodic, whereas recalling the name of a famous gallery would likely be semantic (unless, for example, producing this information relied upon a memory of reading a brochure).

Nuxoll and Laird [14] demonstrated that an episodic store can afford an intelligent agent a multitude of cognitive capabilities. For example, an agent that recalls the results of actions taken previously in situations similar to its current state may learn to predict the immediate consequences of future actions (i.e. action modeling). In this paper, we extend this previous work, augmenting and refining the underlying implementation and providing a theoretical and empirical analysis of the algorithms and data structures necessary to support effective and efficient episodic memory.

Episodic memory research is closely related to studies in case-based reasoning (CBR). The goal of CBR is to optimize task performance given a case-base, where each case consists of a problem and its solution [6]. In CBR systems, case structure is typically pre-specified, case-base size is either fixed or grows at a limited rate, and the cases usually do not have any inherent temporal structure. In contrast, an episodic store grows monotonically with experience, accumulating snapshots of an agent’s experiences over time. An agent endowed with this

memory can retrieve relevant episodes to facilitate reasoning and learning based upon prior events.

In Section 2, we characterize episodic memory and introduce performance goals for its effective use in computational systems. We then compare our goals for episodic memory with related work. In Section 3, we present a brief overview of Soar-EpMem, the integration of episodic memory with Soar 9 [8], together with the domain we will use for evaluation. In Section 4, we present the details of Soar-EpMem, formally characterize its performance, and empirically evaluate its performance on a complex domain (>2500 features) over one million episodes. Earlier versions of Soar-EpMem [14] emphasized the cognitive capabilities episodic memory affords, while this work describes a completely new, robust, efficient, more complete implementation, together with the formal characterization and empirical evaluation.

## 2 Characterizing Episodic Memory

In this section, we first enumerate the relevant functional and implementation requirements for episodic memory. We then contextualize episodic memory within related case-based reasoning work. Finally, we analyze an existing task-independent episodic memory system apropos efficient implementation.

### 2.1 Episodic Memory Constraints

Episodic memory is distinguished from other memory systems by a set of functional requirements [13]. First, episodic memory is *architectural*: it is a functionality that does not change from task-to-task. Second, episodic memory is *automatic*: memories are created without a deliberate decision by the agent. Because memories are experiential, there is some temporal relationship between episodes. Furthermore, in contrast to most case-based reasoning systems, the set of features that can occur in an episodic memory is not pre-specified, but reflects whatever data is available via agent sensing and internal processing. Also, there is not a pre-specified subset of the episode that is used for retrieval nor is there a pre-specified subset that is the result, so that the complete episode is reconstructed during retrieval.

For this work, we adopt two additional restrictions on episodic memory:

1. Stored episodes do not change over time: memories do not decay and are not removed from the episodic store (no forgetting).
2. The goal of retrieval is to find the episode that is the Nearest Neighbor (NN) to the cue based upon qualitative matching (using recency of the episode as a tie-breaking bias).

This specification imposes challenging complexity bounds: the lack of episode dynamics dictates a monotonically increasing episodic store and capturing full agent state over time requires storage at least *linear* in state *changes*.

## 2.2 Related Case-Based Reasoning Work

Efficient NN algorithms have been studied in CBR for qualitative and quantitative retrieval [11] [21] [24]. The underlying algorithms and data structures supporting these algorithms, however, typically depend upon a relatively small and/or static number of case/cue dimensions, and do not take advantage of the temporal structure inherent to episodic memories.

Considerable work has been expended to explore heuristic methods that exchange reduced competency for increased retrieval efficiency [19], including refined indexing [2] [3], storage reduction [25], and case deletion [17]. Many researchers achieve gains through a two-stage cue matching process that initially considers *surface* similarity, followed by *structural* evaluation [4]. In this work, we take advantage of this approach and apply it to a monotonically growing, task-independent episodic memory.

The requirement of dealing with time-oriented problems has been acknowledged as a significant challenge within the CBR community [1], motivating work on temporal CBR (T-CBR) systems [16], and research on the representation of and reasoning about time-dependent case attributes [5], as well as preliminary approaches to temporal case sequences [12] [18]. However, existing T-CBR work does not deal with accumulating an episodic store, nor does it take advantage of temporal structure for efficient implementations.

## 2.3 EM: A Generic Memory Module for Events

EM [22] is a generic store to support episodic memory functionality in a variety of systems, including planning, classification, and goal recognition. EM is an external component with an API, wherein host systems must implement a thin interface layer. The term “episode” in EM defines a sequence of actions with a common goal and is represented as a triple: context (“general setting” of the episode), content (ordered set of the events that make up the episode), and outcome (a domain/task-specific evaluation of the result of the episode). Though meaningful in systems like planners, it may be difficult to pre-define action sequences and outcome evaluation functions for long-living agents that must contend with multiple, possibly novel, tasks.

EM queries are partially defined episodes and a single evaluation dimension. EM utilizes a two-stage evaluation scheme, whereby a constant number (5) of potential matches are found, which are then compared using a relatively expensive semantic matcher. While Tecuci and Porter have shown results for learning in short (250 episode), single-task domains, it is unclear whether the underlying algorithms and data structures will scale to agents with many orders of magnitude more episodes.

## 3 Integration of Episodic Memory in Soar

This section provides an introduction to the integration of episodic memory with Soar 9 [8], followed by an overview of the task we will use for evaluation.

### 3.1 Soar

Soar is a cognitive architecture that has been used extensively for developing AI applications and cognitive models. One of Soar’s main strengths has been its ability to efficiently represent and bring to bear large bodies of symbolic knowledge to solve diverse problems using a variety of methods [10].

Although Soar has many components, the most relevant aspect of its design to episodic memory is that it holds all of its short-term knowledge in its working memory (WM). Working memory contains an agent’s dynamic internal state, including perceptual data, situational awareness, current goals and intentions, and motor commands. By recording the contents of working memory, episodic memory can capture an agent’s history of experience.

Soar’s working memory is implemented as a directed, connected graph of working memory elements (WMEs). Each WME is a triple: *identifier* (a node or vertex), *attribute* (a link or edge), and *value* (a node or terminal value). Working memory is a set: no two WMEs can have the same identifier, attribute, and value.

### 3.2 Episodic Memory Integration

Figure 1 depicts the high-level integration of episodic memory with Soar. Episodes consist of snapshots of working memory and are automatically *stored* in the episodic store. Episodes are retrieved when an agent deliberately creates a cue, at which point Soar searches the store for candidate episodes, ranks them with respect to the cue (*cue matching*), selects the best match, and then *reconstructs* the episode in working memory (in a special area so that it does not overwrite existing working memory structures, nor is the retrieved episode confused with the agent’s current experience).

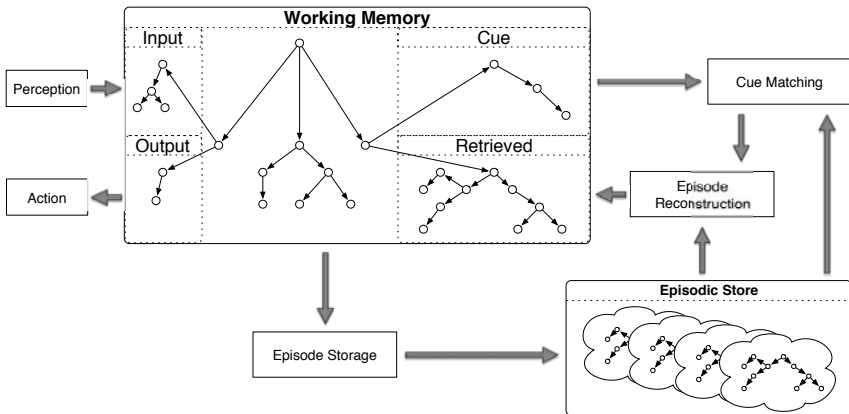


Fig. 1. Soar-EpMem Architecture

Soar’s representation of working memory as an arbitrary graph structure has significant implications for the underlying implementation of episodic memory. A simpler representation, such as a vector or propositional representation, would make it possible to develop a simpler and faster implementation of episodic memory, but at significant cost in expressability and generality. The underlying implementation of episodic memory is independent of other details of Soar and should generalize to other architectures with graph-based representations of their dynamic data.

Soar agents interact with real-world, dynamic environments and these agents have real-time constraints on reactivity. Based on our own experience with real-time agents, Soar must execute its primitive cycle in 50-100ms to maintain reactivity in real-world agents. This bound is easily achieved for Soar’s non-episodic memory components and puts a limit on how much time can be spent on storage, which can occur every cycle. However, there is more flexibility in retrieval, which can occur in parallel with Soar’s other processing and can still be useful if it takes small multiples of Soar’s primitive cycle time. Empirical evidence suggests that a retrieval time of approximately 500ms is necessary to be useful in real world applications.

### 3.3 Evaluation Domain

Our evaluation of Soar-EpMem is based on the TankSoar domain. TankSoar is a pre-existing domain that has been used extensively in evaluating other aspects of Soar and was used in the original episodic memory research in Soar [13]. In TankSoar, each Soar agent controls an individual tank that moves in a discrete 15x15 two-dimensional maze. Agents have only limited sensing of the environment and their actions include turning, moving, firing missiles, raising shields, and controlling radar. A TankSoar agent has access to a rich set of environmental features through its senses, including smell (shortest-path distance to the nearest enemy tank), hearing (the sound of a nearby enemy), path blockage, radar feedback, and incoming missile data.

TankSoar includes an agent named *mapping-bot* that builds up an internal map of the environment as it explores. The *mapping-bot* agent’s working memory contains about 2,500 elements. Over 90% of these elements comprise a static map of its environment. A large proportion of the remaining WMEs (usually 70-90%) are related to perception and they typically change within one episode. For the experiments described below, a new episode was stored every time an action was performed in the environment, which is approximately every primitive decision cycle in Soar. The properties of this agent, especially the large working memory and the large number of WMEs changing per episode, make TankSoar an atypically stressful domain for episodic memory experimentation.

The tests were run on an Intel 2.8GHz Core 2 Duo processor and 4GB RAM. The Soar-EpMem episodic store was managed using version 3 of the SQLite in-process relational database engine [20]. The tests described below involved one million *mapping-bot* episodes, averaged over ten trials.

## 4 Soar-EpMem Structure and Evaluation

In this section, we present Soar-EpMem together with its evaluation. We begin with a description of global data structures that summarize the structures found in working memory. These global structures greatly decrease the amount of storage and processing required for individual episodes. The remaining subsections describe the different phases of episodic memory processing. The first is *storage* of episodic memories, the second is *cue matching* to find a stored episodic memory, and the final is *episodic reconstruction*, which involves finding all of the components of an episode and adding it to Soar’s working memory.

The design of Soar-EpMem was motivated by the need to minimize the growth in processing time for all of these operations as the number of episodes increases. However, over long agent lifetimes, *cue matching* has the greatest growth potential and the overall design is meant to minimize the time required for that operation, without significantly impacting required memory or the time required for the other operations.

### 4.1 Global Episodic Memory Structures

In order to eliminate duplicate representations of working memory elements and speed *cue matching*, a global structure can be maintained that represents all the structures that have existed in working memory. A naïve storage representation would explicitly define an episode as a “bag” of pointers to such a global structure (see Fig. 2, left). Termed an “instance” representation by Nuxoll and Laird [14], such an approach requires time and storage linear in the average number of working memory elements per episode.

Our design for Soar-EpMem takes advantage of the temporal structure of episodes, namely that one episode usually will differ from the previous (and next) episode only in a relatively small number of features. Thus, throughout its design, Soar-EpMem attempts to process *only changes* to working memory instead of the complete episode. As a result, storing an episode involves only noticing which elements have been added and which elements have been removed from working memory, building up ranges of when working memory elements existed. Termed an “interval” representation by Nuxoll and Laird, episodes are implic-

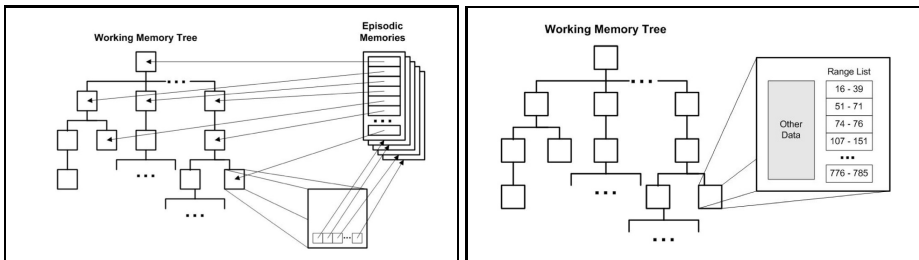


Fig. 2. Global Episodic Memory Structures [13]: Instance (left), Interval (right)



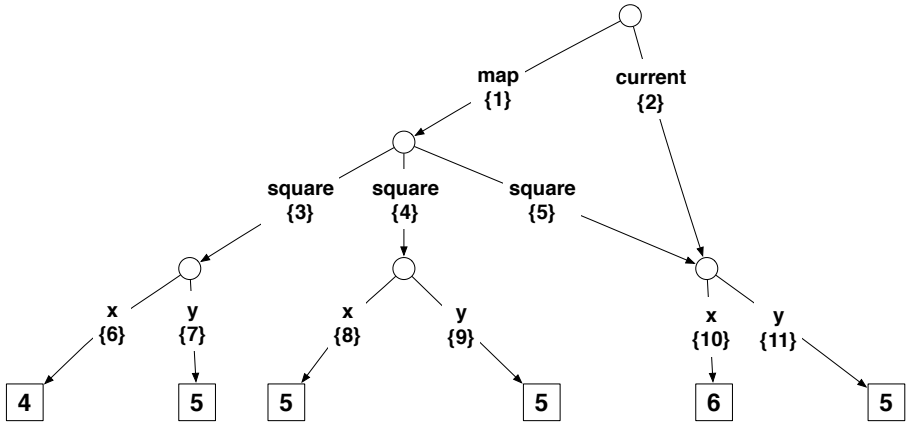


Fig. 3. Working Memory Example

itly represented by associating valid temporal ranges with a global structure (see Fig. 2, right). With this approach, *episode storage* is achieved in time and space linear with respect to the *changes* in the agent’s working memory [13].

In Nuxoll’s original work [13], the graph structure of working memory was simplified such that every attribute of an object was unique and that the overall structure was a tree. Fig. 3 shows a simple working memory that violates both of these assumptions. There are multiple squares ( $\{3\}$ ,  $\{4\}$ ,  $\{5\}$ ) in the map ( $\{1\}$ ), and a “square” ( $\{5\}$ ) and “current” ( $\{2\}$ ) edge share a common descendant. Under the assumption that working memory was a tree, Nuxoll’s system built up a global structure (termed a *Working Memory Tree*) of all unique structures that had ever occurred in working memory. Using a tree instead of a graph for modeling working memory greatly simplified Nuxoll’s implementation and was sufficient for him to explore applications of episodic memory; however these simplifications make it impossible to correctly search and reconstruct episodes based upon relational working memory structures, which is necessary for many real-world applications. To correct this deficiency, Soar-EpMem implements a new global structure, the *Working Memory Graph* (WMG), which captures all information necessary for a faithful episode representation.

## 4.2 Episode Storage

The Working Memory Graph captures all distinct edges that have occurred in Soar’s working memory. By associating valid temporal intervals (see Fig. 2, right) with the unique ids of each edge (such as  $\{1\}$ ,  $\{2\}$ , etc.), we implicitly define structure for individual episodes. *Episode storage* is the process of efficiently recording the start/end of these intervals.

In Soar-EpMem, interval ranges are started by executing the following algorithm for every element of working memory:

1. if element already points to the WMG, ignore
2. else:
  - (a) if a corresponding edge does not exist in the WMG, add it
  - (b) point the element to the corresponding WMG edge
  - (c) start a new interval at the pointed WMG edge

By ignoring elements with existing pointers (step 1), we process only *new* working memory elements. Later, following an element’s removal from working memory, Soar-EpMem records the end of the corresponding edge interval. Thus our implementation only stores element *changes*.

Using this approach, Soar-EpMem *storage* time across one million episodes of *mapping-bot* remained approximately constant, requiring 1.48-2.68 ms/episode. Total episodic store size ranged from 625-1620 MB, averaging between 0.64 and 1.66 KB/episode (respectively).

### 4.3 Cue Matching

Episodic cues take the form of an acyclic graph, partially specifying a subset of an episode. For example, an agent can create a cue of a position in the map to recall what it sensed the last time it was in that position.

Episode retrieval proceeds as follows:

1. 2-Phase Nearest Neighbor Cue Matching
  - (a) Identify candidate episodes based upon *surface* cue analysis.
  - (b) Perform *structural* cue analysis on *perfect surface* matches.
2. Select and reconstruct the best matching episode (if it exists).

Candidate episodes are defined as containing at least one cue leaf element. Our *surface* evaluation function returns the “balanced” sum of match cardinality (number of matching leaf nodes) and feature weighting [15].

Candidate episodes with perfect match cardinality are considered *perfect surface matches* and are submitted to a second phase of *structural* graph-match. The graph-match algorithm implements standard CSP backtracking. The best matching episode is either the most recent *structural* match or the highest scoring *surface* match.

Soar-EpMem efficiently iterates over candidate episodes by implementing Nuxoll and Laird’s [13] interval searching algorithm. The major insight of this algorithm is that a candidate match score changes only at the endpoints of episode element intervals. For example, consider Fig. 4. The dashed vertical line at episode 6 indicates the time point of maximum coverage (i.e. maximum match cardinality). If we begin from the most recent time point (time point 12) and consider each candidate episode, we will examine 7 episodes before arriving at this peak. However, after scoring episode 8, there will be no change in coverage until time point 6: considering episodes between endpoints is redundant. Thus

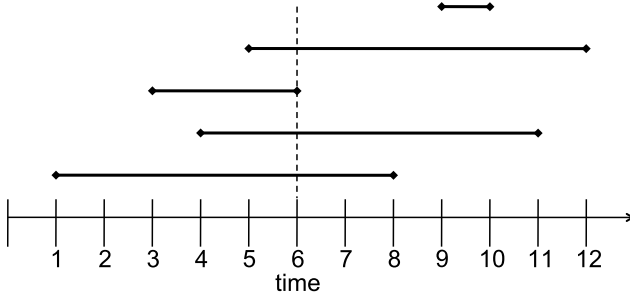


Fig. 4. Interval Search Example

if we just walk interval endpoints, incrementally updating match score, we can achieve significant computational savings. By *only processing changes*, candidate episode iteration achieves time linear in the relevant element *changes* (as opposed to linear in the number of episodes).

As an optimization, Soar-EpMem will cease interval search upon reaching a *structural* match. In Fig. 4, however, there is no perfect *surface* match (since no episode is covered by all ranges). Thus, if we assume uniform feature weighting, interval search will return episode 6 after processing all 10 endpoints.

Soar-EpMem efficiently implements Nuxoll and Laird’s interval search algorithm by maintaining B+-tree indexes of all temporal interval endpoints (one for interval start, one for interval end), keyed on episode nodes. Walking a node’s endpoints in descending order of time entails finding the most recent time of interest (log time with respect to the number of endpoints), and walking the leaf nodes in order (constant time per endpoint). To process a multi-node cue, we maintain parallel B+-tree leaf pointers for each cue node. All pointers within a B+-tree are stored in priority queues (keyed on endpoint value). We then efficiently walk endpoints as described above.

Because Soar’s working memory is implemented as a graph, cue leaf nodes do not uniquely identify an edge in the Working Memory Graph. For example, consider the cue in Fig. 5, left (relating to working memory as illustrated in Fig. 3). The internal cue node representing a “square” can be satisfied by any of the three working memory squares ( $\{3\}$ ,  $\{4\}$ ,  $\{5\}$ ). Thus, the problem of satisfying a leaf node is tantamount to satisfying the sequence of nodes that leads to (and includes) the leaf node. To continue the example, we can formally express the satisfaction of the two cue leaf nodes with the following monotonic, disjunctive normal form (DNF) boolean statements (respecting ids in Fig. 3):

$$\begin{aligned}
 \text{sat}(x = 4) &:= ( \{1\} \wedge \{3\} \wedge \{6\} ) \\
 \text{sat}(y = 5) &:= ( \{1\} \wedge \{3\} \wedge \{7\} ) \vee \\
 &\quad ( \{1\} \wedge \{4\} \wedge \{9\} ) \vee \\
 &\quad ( \{1\} \wedge \{5\} \wedge \{11\} )
 \end{aligned}$$

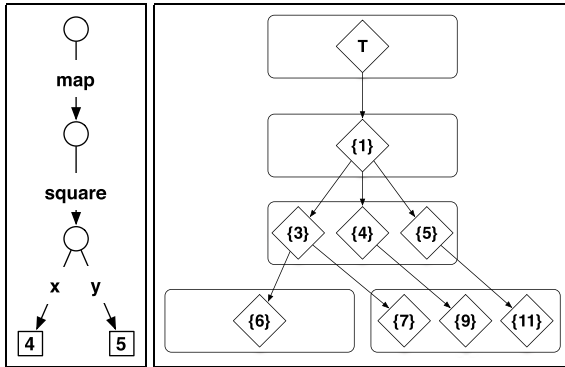


Fig. 5. Retrieval Example: Cue (left), Corresponding DNF Graph (right)

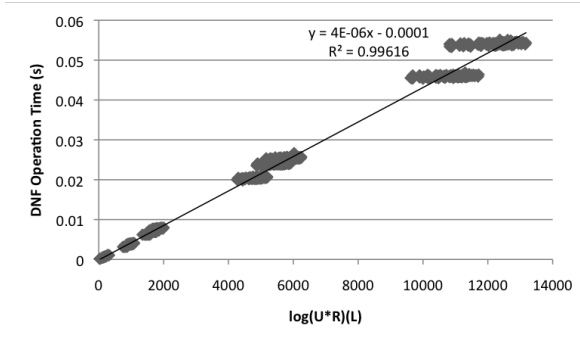
Thus, the problem of efficiently implementing interval search with a Working Memory Graph is analogous to efficiently tracking satisfaction of a set of DNF boolean equations. Soar-EpMem solves this problem by *only processing changes* to a corresponding *DNF graph* (depicted in Fig. 5, right).

The interval search process is initialized while processing the cue. During a breadth-first search, we simultaneously create the priority queue of B+-tree pointers (as discussed above) and the DNF graph. For clarity, a clause in a DNF statement is represented as a root-to-leaf path in the DNF graph. Each literal in the DNF graph (depicted as a diamond) has an associated count value, initialized to 0, representing local satisfaction. A literal count is incremented if its associated cue element node is “active” (i.e. our current state of interval search is within the start/end endpoints of the node) OR its parent has a counter value of 2. The root literal (shown with a special id of “T”) is initialized with count 2 (thus initializing the count of all direct children to 1). If at least one leaf literal is satisfied (i.e. has count value 2), the clause must be satisfied. We additionally maintain a global match score, initialized to 0.

At each endpoint in the interval search algorithm, exactly one literal is activated or de-activated (depending on whether we encounter an end/start). If the literal is part of an internal cue node, this change may entail recursive propagation to child literals. If during propagation we alter clause satisfaction, we modify the global match score. Thus, we extend endpoint iteration to track *only changes* in boolean satisfaction of the DNF graph and, by extension, modifications of candidate match score.

To compare Soar-EpMem *cue matching* performance with theoretical bounds, we developed the following model to reflect the effects of operational algorithms and data structures:

$$\begin{aligned}
 \text{Cue Match} &= \text{DNF} + \text{Interval Search} + \text{Graph Match} \\
 \text{DNF} &= (X_1)(\log_2[U * R])(L) \\
 \text{Interval Search} &= (X_2)(1/T)(\text{Distance})(\Delta)
 \end{aligned}$$

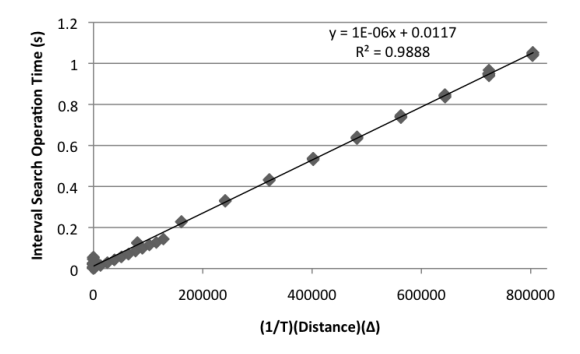


**Fig. 6.** *Cue Matching* DNF Regression Data

The constants in the equations ( $X_1$ ,  $X_2$ ) reflect linear scaling factors for a given computer. To derive these values for our experimental setup, we performed 100 isolated executions of primitive operations (*DNF* and *Interval Search*) on data collected from 10 trials of *mapping-bot* data at 10 time points (100K, 200K, ... 1M). We collected the necessary episode statistics (described below) and performed linear regressions to fit data points for 15 different queries. Low performance timers (resolution was  $1\mu\text{s}$ ) caused most model noise.

The *Cue Match* operation comprises *DNF* construction and *Interval Search*. The former is linearly dependent upon the logarithmic growth of the average number,  $U$ , of historically unique internal and leaf nodes multiplied by  $R$ , the total number of stored intervals, as well as linearly dependent upon  $L$ , the number of literals associated with the cue nodes. In our tests (see Fig. 6), we found  $X_1$  to be  $4.33\mu\text{s}$  ( $R^2=0.996$ ). In experiments with *mapping-bot* to one million episodes, results depended greatly on the cue. For all cues that did not reference “squares” on the agent’s internal map, *DNF* operation time was constant and below 8 ms. Cues containing references to map “squares” (and thus referring to over 250 underlying structures) brought this upper bound to 55.1 ms.

The *Interval Search* operation is expressed as a proportion of relevant cue node intervals.  $T$  represents the total number of episodes recorded. *Distance* represents the temporal difference between the current episode and the best match.  $\Delta$  represents the total number of intervals relevant to the cue. Intuitively, the farther back in time we must search for an episode, the more intervals we must examine. This ratio could be re-written as the product the minimal relative co-occurrence probability of the cue nodes, and the total number of changes experienced to date by these cue nodes. In our tests (see Fig. 7), the  $X_2$  constant was  $1.29\mu\text{s}$  ( $R^2=0.989$ ). We found absolute operation times depended greatly on the supplied cue. For cues that did not compel distant searches, *Interval Search* was constant with an upper bound of 2.5 ms. With cues crafted to force a linear scan of the episodic store, time increased linearly to a maximum of 1.03 seconds over one million episodes.



**Fig. 7.** *Cue Matching* Interval Search Regression Data

Since the linear factors,  $L$  and  $\Delta$ , grow proportionally to *changes* in agent working memory, the first phase of Soar-EpMem *cue matching* achieves the lower bound of growing *linearly* with agent *changes*.

The *Graph Match* operation, however, is much more difficult to characterize. CSP backtracking depends upon cue breadth, depth, structure (such as shared internal cue nodes), and corresponding candidate episodes, but can be combinatorial in the worst case (though our two-phase matching policy attempts to minimize this cost). We have not extensively evaluated this component, but we expect a studied application of heuristic search will effectively constrain graph-match in the average case.

#### 4.4 Episode Reconstruction

Given the temporal id of an episode, *reconstruction* in Soar-EpMem can be summarized by the following two steps:

1. Collect contributing episode elements
2. Add elements to working memory

Given the episode elements, adding WMEs to Soar is a straight forward process that takes advantage of Soar’s existing architectural primitives and thus we focus on the former step.

Collecting episode elements in an “interval” representation (see Fig. 2, right) is tantamount to an interval intersection query: collect all elements that started before and ended after time  $t$ . To facilitate efficient *episode reconstruction*, Soar-EpMem maintains a Relational Interval Tree (RI-tree) [7]. An RI-tree is a mapping of the interval tree data structure onto Relational Database Management System (RDBMS) B+-tree indexes and SQL queries. As with a standard interval tree, intersection queries execute in logarithmic time with respect to the number of stored intervals.

Excluding system-specific step 2 above, we developed the following model for *episode reconstruction* performance in Soar-EpMem:

$$\begin{aligned} \text{Reconstruction} &= \text{RI-tree} + \text{Collect} \\ \text{RI-tree} &= (X_3)(\log_2 R) \\ \text{Collect} &= (X_4)(M)(1 + \log_2 U) \end{aligned}$$

To validate our model we performed 100 isolated executions of primitive operations (*RI-tree* and *Collect*) on the same data collected for *cue-matching* (10 trials, *mapping-bot*, 10 time points from 100K to 1M episodes). We collected the necessary statistics (described below) for 50 episodes selected randomly (5 per 10,000 episodes through the first 100,000 episodes of execution) and performed linear regressions to fit data points.

Total time for *episode reconstruction* is the sum of two operations: *RI-tree* and *Collect*. *RI-tree* refers to the process of extracting pertinent intervals from the Relational Interval Tree. The logarithmic dependent variable,  $R$ , refers to the total number of ranges in the RI-tree structure. In our experiments, the  $X_3$  constant was  $2.55\mu\text{s}$  (over 70%  $R^2$ ). After one million episodes, we recorded the upper bound of *RI-tree* operation time as 0.1 ms.

The *Collect* operation refers to cross-referencing pertinent episode intervals with structural information in the Working Memory Graph. This process depends upon the average number,  $U$ , of historically unique internal and leaf nodes, as well as the number of elements,  $M$ , comprising the episode to be reconstructed. With *mapping-bot* we regressed an  $X_4$  value of  $1.6\mu\text{s}$ . Because episode size does not vary greatly in *mapping-bot* (2500-2600 elements, typically), the dominating linear factor,  $M$ , highlighted noise in the experimental data and thus  $R^2$  was 73%. After one million episodes, we recorded an upper bound of 22.55 ms for the *collection* operation with episodes ranging from 2521-2631 elements.

If we assume a constant or slowly growing average episode size, the  $M$  factor can be considered a constant and thus *Reconstruction* becomes the linear sum of logarithmic components  $R$  and  $U$ . Both  $R$  and  $U$  increases result from *changes* in agent working memory. Thus, under these assumptions, Soar-EpMem *episode reconstruction* achieves the lower bound of growing *linearly* with agent *changes*.

## 5 Conclusion

In this paper, we presented an implementation of a graph-based, task-independent episodic memory and characterized the associated computational challenges. We provided formal models of the costs associated with the different phases of episodic processing and provided empirical results over one million episodes.

In this work we applied efficient data structures and algorithms to limit episodic computation, while still guaranteeing a best-match retrieval. Thus, we consider that a typical cue is one for which retrieval does not require a linear scan of the episodic store. Table 1 summarizes typical empirical results for *mapping-bot* over one million episodes. *Storage* time remains nearly constant, and well below the bound of 50ms, with linear growth in storage. *Cue matching* time is within desired bounds, but suffers linear growth in the atypical case (with a

**Table 1.** Soar-EpMem Typical Operation Costs: *mapping-bot*

episodes	<i>storage</i>	<i>cue matching</i>	<i>reconstruction</i>	total
1,000,000	2.68ms, 1620MB	57.6ms	22.65ms	82.93ms

maximum observed cost of 1.03 sec.). Episodic *reconstruction* time is dominated by episode size but falls within desired bounds for over 2500 features. Although these results achieve our initial performance goals, there is still much to be done:

- **Additional Empirical Evaluation.** We need to expand to additional domains and establish a set of test cases for episodic memory systems. TankSoar is a stressful test, useful for initial exploration and evaluation, but it is artificial and we need a collection of tasks that use episodic memory in a variety of ecologically valid ways.
- **Extended Evaluations.** We also need to explore much longer runs. One million episodes corresponds to approximately fourteen hours of real time. Our goal is to have agents that exist continually for 1 year (42-420 million episodes [9]).
- **Cue Match Bounding.** Although storage and reconstruction are relatively well behaved, the cost of cue matching can be extremely variable depending on the complexity of the cue and the structures in episodic memory. We need to explore alternative or even heuristic graph-matching schemes that provide tighter bounds.

## References

1. Combi, C., Shahar, Y.: Temporal Reasoning and Temporal Data Maintenance in Medicine: Issues and Challenges. *Computers in Biology and Medicine* 27(5), 353–368 (1997)
2. Daengdej, J., Lukose, D., Tsui, E., Beinat, P., Prophet, L.: Dynamically Creating Indices for Two Million Cases: A Real World Problem. In: Smith, I., Faltings, B. (eds.) *EWCBR 1996*. LNCS, vol. 1168, pp. 105–119. Springer, Heidelberg (1996)
3. Fox, S., Leake, D.: Using Introspective Reasoning to Refine Indexing. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 391–399. Morgan Kaufmann, San Francisco (1995)
4. Forbus, K., Gentner, D., Law, K.: MAC/FAC: A Model of Similarity-based Retrieval. *Cognitive Science* 19(2), 141–205 (1995)
5. Jære, M., Aamodt, A., Skalle, P.: Representing temporal knowledge for case-based prediction. In: Craw, S., Preece, A.D. (eds.) *ECCBR 2002*. LNCS, vol. 2416, pp. 174–234. Springer, Heidelberg (2002)
6. Kolodner, J.: An Introduction to Case-Based Reasoning. *Artificial Intelligence Review* 6(1), 3–34 (1992)
7. Kriegel, H., Pötke, M., Seidl, T.: Managing Intervals Efficiently in Object-Relational Databases. In: *Proceedings of the 26th International Conference on Very Large Databases*, pp. 407–418. Morgan Kaufmann, San Francisco (2000)



8. Laird, J.E.: Extending the Soar Cognitive Architecture. In: Proceedings of the 1st Conference on Artificial General Intelligence, pp. 224–235. IOS Press, Amsterdam (2008)
9. Laird, J.E., Derbinsky, N.: A Year of Episodic Memory. In: Workshop on Grand Challenges for Reasoning from Experiences, 21st International Joint Conference on Artificial Intelligence (2009)
10. Laird, J.E., Rosenbloom, P.: The Evolution of the Soar Cognitive Architecture. *Mind Matters: A Tribute to Allen Newell*, pp. 1–50. Lawrence Erlbaum Associates, Inc., Mahwah (1996)
11. Lenz, M., Burkhard, H.: Case Retrieval Nets: Basic Ideas and Extensions. In: Görz, G., Hölldobler, S. (eds.) KI 1996. LNCS, vol. 1137, pp. 227–239. Springer, Heidelberg (1996)
12. Ma, J., Knight, B.: A Framework for Historical Case-Based Reasoning. In: Ashley, K., Bridge, D. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 246–260. Springer, Heidelberg (2003)
13. Nuxoll, A.: Enhancing Intelligent Agents with Episodic Memory. PhD Dissertation, University of Michigan, Ann Arbor (2007)
14. Nuxoll, A., Laird, J.E.: Extending Cognitive Architecture with Episodic Memory. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, pp. 1560–1564. AAAI Press, Vancouver (2007)
15. Nuxoll, A., Laird, J.E., James, M.: Comprehensive Working Memory Activation in Soar. In: Proceedings of the 6th International Conference on Cognitive Modeling, pp. 226–230. Lawrence Erlbaum Associates, Inc., Mahwah (2004)
16. Patterson, D., Galushka, M., Rooney, N.: An Effective Indexing and Retrieval Approach for Temporal Cases. In: Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference, pp. 190–195. AAAI Press, Vancouver (2004)
17. Patterson, D., Rooney, N., Galushka, M.: Efficient Retrieval for Case-Based Reasoning. In: Proceedings of the 16th International Florida Artificial Intelligence Research Society Conference, pp. 144–149. AAAI Press, Vancouver (2003)
18. Sánchez-Marré, M., Cortés, U., Martínez, M., Comas, J., Rodríguez-Roda, I.: An approach for temporal case-based reasoning: Episode-based reasoning. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS, vol. 3620, pp. 465–476. Springer, Heidelberg (2005)
19. Smyth, B., Cunningham, P.: The Utility Problem Analysed: A Case-Based Reasoning Perspective. In: Smith, I., Faltings, B. (eds.) EWCBR 1996. LNCS, vol. 1168, pp. 392–399. Springer, Heidelberg (1996)
20. SQLite, <http://www.sqlite.org>
21. Stottler, R., Henke, A., King, J.: Rapid Retrieval Algorithms for Case-Based Reasoning. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence, pp. 233–237. Morgan Kaufmann, San Francisco (1989)
22. Tecuci, D., Porter, B.: A Generic Memory Module for Events. In: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference, pp. 152–157. AAAI Press, Vancouver (2007)
23. Tulving, E.: *Elements of Episodic Memory*. Clarendon Press, Oxford (1983)
24. Wess, S., Althoff, K., Derwand, G.: Using k-d Trees to Improve the Retrieval Step in Case-Based Reasoning. In: Wess, S., Althoff, K., Richter, M. (eds.) EWCBR 1993, vol. 837, pp. 167–181. Springer, Heidelberg (1994)
25. Wilson, D., Martinez, T.: Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning* 38(3), 257–286 (2000)

# Integration of a Methodology for Cluster-Based Retrieval in jColibri

Albert Fornells<sup>1</sup>, Juan Antonio Recio-García<sup>2</sup>, Belén Díaz-Agudo<sup>2</sup>,  
Elisabet Golobardes<sup>1</sup>, and Eduard Fornells<sup>1</sup>

<sup>1</sup> Grup de Recerca en Sistemes Intel·ligents

La Salle - Universitat Ramon Llull

Quatre Camins 2, 08022 Barcelona, Spain

{[afornells](mailto:afornells@salle.url.edu), [elisabet](mailto:elisabet@salle.url.edu), [efornells](mailto:efornells@salle.url.edu)}@salle.url.edu

<sup>2</sup> Department of Software Engineering and Artificial Intelligence,

Universidad Complutense de Madrid, Spain

[jareciog@fdi.ucm.es](mailto:jareciog@fdi.ucm.es), [belend@sip.ucm.es](mailto:belend@sip.ucm.es)

**Abstract.** One of the key issues in Case-Based Reasoning (CBR) systems is the efficient retrieval of cases when the case base is huge and/or it contains uncertainty and partial knowledge. Although many authors have focused on proposing case memory organizations for improving the retrieval performance, there is not any free open source framework which offers this kind of capabilities. This work presents a plug-in called *Thunder* for the jCOLIBRI framework. Thunder provides a methodology integrated in a graphical environment for managing the case retrieval from cluster based organizations. A case study based on tackling a Textual CBR problem using Self-Organizing Maps as case memory organizing technique is successfully tested.

**Keywords:** CBR Tools, Cluster Based Memory, Case Memory Organization, Soft Case-Based Reasoning, Textual CBR, jCOLIBRI.

## 1 Introduction

Now that Case-Based Reasoning (CBR) has become a mature and established technology, there is a necessity for methodologies and tools to build CBR systems taking into account the accumulated practical experience of applying CBR techniques to real-world problems. In the CBR community, different tools have emerged: CBR\*Tools [1], Orange (the Open Retrieval engine from empolis.com) [2], MyCBR [3], IUCBRF [4], jCOLIBRI [5]. Each one of these tools offer different features, and a variety of different methods for organizing, retrieving, utilizing and indexing the knowledge retained in past cases, however none of them provide a global support for different types of systems and knowledge sources. jCOLIBRI is becoming one of the most important tools due to its open architecture that allows other people to contribute reusable methods and templates of systems. Thanks to these contributions, its library offers a variety of methods big enough to design different types of systems according to the specific domain and application requirements [5].

One of the problems to solve when dealing with real world problems is the efficient retrieval of cases when the case base is huge and/or it contains uncertainty and partial knowledge. There are many examples of domains and applications where a huge amount of data arises, for example, image processing, personal records, recommender systems, textual sources, and many others. Many authors have focused on proposing case memory organizations to improve the retrieval performance. For example, there are different proposals to manage huge case memories organized in clusters such as in [6,7]. However none of the existing tools has incorporated capabilities to efficiently manage big case bases. This paper makes a first attempt to fill this gap providing a plugging called *Thunder* for the jCOLIBRI framework. *Thunder* allows CBR experts to manage case memories organized in clusters and incorporates a case memory organization model based on Self-Organizing Maps (SOM) [8] as a clustering technique. Clustering is featured by grouping cases according to their similarities and represent each one of these groups by prototypes. Thus, the retrieve phase carries out a selective retrieval focused on using only the subset of cases potentially similar to the new case to solve. The new case retrieval procedure consists in (1) selecting the most suitable group of cases comparing the input case with the prototypes and, (2) comparing the new input case with the cases from the selected clusters. The benefits of such approach are both the reduction of computational time and improved robustness with uncertain data. Nevertheless, some open issues still remain such as to what extent the accuracy rate is degraded due to the cluster-based retrieval, and furthermore, how many clusters and cases should be used according to given requirements of computational time and accuracy degradation. Both problematical aspects were successfully solved in a methodology [9] based on a procedure for selecting the most suitable configuration according to the user requirements and the data complexity [10].<sup>1</sup> Because this issue is crucial for the success of the retrieval, *Thunder* gives support to this methodology.

The paper is organized as follows. Section 2 reviews briefly the methodology and the jCOLIBRI framework. Section 3 describes the *Thunder* plug-in. Section 4 tests the plug-in using SOM as an organization technique in a textual domain. Finally, section 5 ends with the conclusion and further research.

## 2 Background Work

The work described in this paper is based on two previous research lines. The first one concerns about a methodology for selecting the most suitable case retrieval strategy from clustered cases memories [9], while the second research work involves the development of a framework (jCOLIBRI) for supporting the building of CBR applications [5]. This section describes both previous works to provide the required background to understand the integration of cluster-based retrieval processes in jCOLIBRI.

---

<sup>1</sup> The data complexity refers to the class separability and the discriminant power of features, and not about its representation in the case memory.

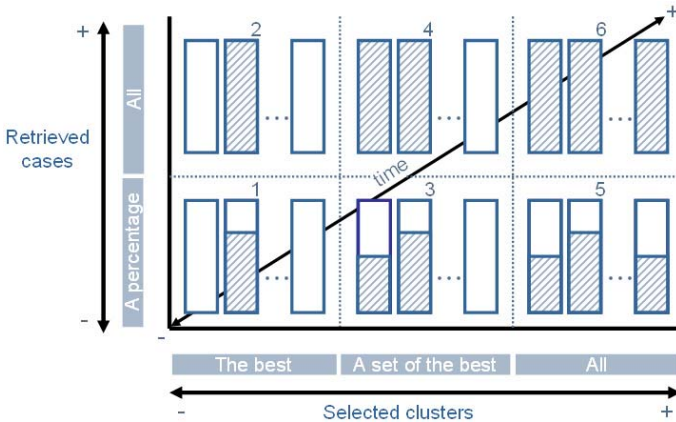


Fig. 1. Strategy map for the taxonomy of cluster-based retrieval strategies

### 2.1 Case Retrieval from a Clustered Case Memory

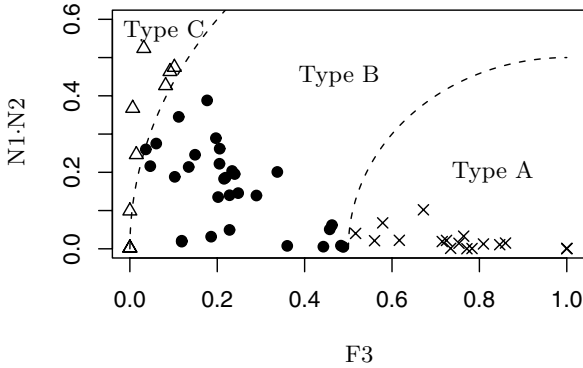
This section summarizes the methodology to characterize the case retrieval strategies from clustered case memories [9]. The characterization is based on the sequential building of three innovative elements: (1) the *strategy map*, (2) the *complexity map*, and (3) the *scatter plot*. At the end of the process, the user is able to take advantage of this analysis to select the strategy that meets best its requirements in terms of accuracy and computational resources through the next steps: (1) to determine the complexity type of the dataset, (2) to decide the minimal and maximal computational time improvement desired and, (3) to select the most suitable configuration according to the impact in the accuracy.

#### The Strategy Map

The characterization of strategies starts by identifying the different ways in which data can be retrieved. The strategy map is a taxonomy that characterizes any case retrieval from a clustered case memory (see Fig. 1). The x-axis depicts the number of clusters selected (*‘The most similar’, ‘A set of the most similar’, or ‘All’*) and the y-axis depicts the number of cases retrieved from each cluster (*‘A percentage’ or ‘All’*). Enclosing the range of cases to explore implies a decrease in computational time to the detriment of the accuracy. Areas 1, 2, and 3 mainly include strategies that reduce drastically the computational time whereas areas 4, 5, and 6 correspond to conservative strategies.

#### The Complexity Map

Once the strategies are characterized, we should proceed by analyzing their behavior over a large set of problems. In this sense, the characterization of the datasets, by means of their complexity, can provide us useful guidelines for analyzing the performance of the cluster-based strategies. Previous studies have



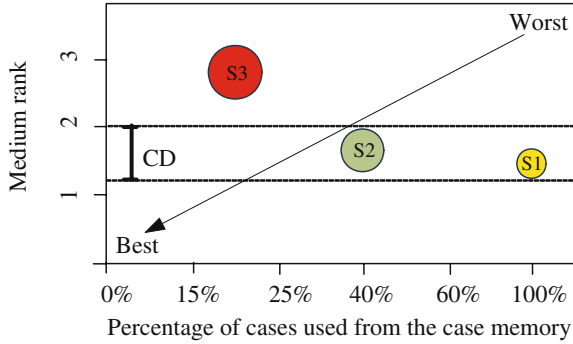
**Fig. 2.** The complexity map determines the complexity level of a dataset analyzing the geometry of the class boundary. Figure shows the classification of 56 datasets selected from UCI Repository [12] in [9].

demonstrated the capability of such measures to explain the intrinsic complexity of the dataset and relate it to the classifier's performance [11].

We base our characterization on three complexity metrics [10] that define a complexity map depicted in three areas A, B, and C, ranging from lower to higher complexity (see Figure 2). A low value of these measures indicates high class separability, while a high value is usually related to high classifier errors and thus, high complexity. The feature efficiency ( $F3$ ) defines to what degree each feature individually takes part in the class separability. The higher the value, the greater the power of class discrimination. The length of the class boundary ( $N1$ ) and the intra/inter class nearest neighbor distances ( $N2$ ) are measures that take into account the class separability. The last two metrics are combined through their product ( $N1 \cdot N2$ ) to emphasize their behaviors.

## The Scatter Plot

The scatter plot is a visual and intuitive way for comparing the strategies' performance -in terms of accuracy and computational time- as regards to a reference strategy (see Fig. 3). The main benefit of this element is the drastic simplification in comparison to analyze complex and huge numerical tables of results. The x-axis measures the computational time improvement computed as the percentage of cases used with respect to a reference strategy in logarithmic scale. The y-axis refers to the solving capabilities as the rank of each strategy averaged over all datasets using the accuracy rate. The performance of each strategy is drawn as a circumference whose diameter is proportional to the standard deviation of the medium rank. Furthermore, the significant difference between any case retrieval strategy and the reference strategy is evaluated by the *critical distance* (CD) using the Bonferroni-Dunn test [13]. The CD delimits a region where the strategies placed inside do not present significant differences with regards to the



**Fig. 3.** The scatter plot for three case retrieval strategies.  $S_2$  and  $S_3$  obtain better computational time than  $S_1$ , but  $S_2$  maintains the same solving capabilities than  $S_1$ .

reference strategy. Our proposal is to characterize the datasets by complexity and then, analyze separately the strategies' performance with the scatter plot.

Figure 3 shows an hypothetical example of the comparison between several case retrieval strategies. Let  $S_1$  be the reference strategy which uses all cases from all clusters. Let  $S_2$  be a strategy which uses all cases from some clusters. Let  $S_3$  be a strategy which uses all cases from only one cluster.  $S_2$  and  $S_3$  improve drastically the computational time since they only use the 40% and the 20% of cases in comparison with  $S_1$ . The solving capabilities of  $S_2$  can be considered similar to  $S_1$  because  $S_2$  is located inside the CD region, while  $S_3$  is statistically different from  $S_1$ .

## 2.2 The jCOLIBRI Framework

jCOLIBRI framework is a platform for the design and generation of CBR systems that proposes a reference architecture for the development of CBR systems, along with its corresponding implementation [5].

jCOLIBRI is currently a popular tool for the development of research prototypes and for teaching purposes. It allows to include third-party contributions with the newest algorithms developed within the CBR community that increase the value of the platform. Along these contributions we can cite the important number of Textual CBR algorithms [14] and Knowledge Intensive similarity measures [15] included in the framework.

jCOLIBRI was primarily designed as a platform for the implementation of KI-CBR applications thanks to the inclusion and exploiting of ontologies and their associated reasoning capabilities. This kind of CBR applications manage a few but rich-defined cases and jCOLIBRI includes specialized methods to process them [16]. However, this specialization for KI-CBR applications has left aside the development of methods for managing case bases with opposite features: many but simple cases. These CBR systems are defined as Data Intensive because they

reason with an important number of cases and do not apply expensive reasoning mechanisms because of performance issues.

The following section describes how authors have solved this lack of Data Intensive capabilities in the jCOLIBRI framework by including clustering techniques. This addition enables the management of large case bases in an efficient way and implies that jCOLIBRI is now a real choice to consider when developing large-scale CBR applications for the industrial world.

### 3 Thunder: The Clustered Case Memory Support

The jCOLIBRI extension to support cluster-based retrieval processes is named *Thunder*. It has been perfectly embedded in the framework due to its layered architecture. jCOLIBRI splits the problem of case base management in two separate although related concerns: persistency mechanisms through connectors and in-memory organization.

#### 3.1 Connectors

Cases are often derived from legacy databases, thereby existing organizational resources are converted into exploitable knowledge. In order to take advantage of these already existing resources, to facilitate intelligent access to existing information and to incorporate it as seed knowledge in the CBR system (the case base) jCOLIBRI offers a set of connectors to manage persistence of cases. Connectors are objects that *know* how to access and retrieve cases from the storage media and return those cases to the CBR system in a uniform way. Therefore, connectors provide an abstraction mechanism that allows users to load cases from different storage sources in a transparent way. As shown in Figure 4, jCOLIBRI includes

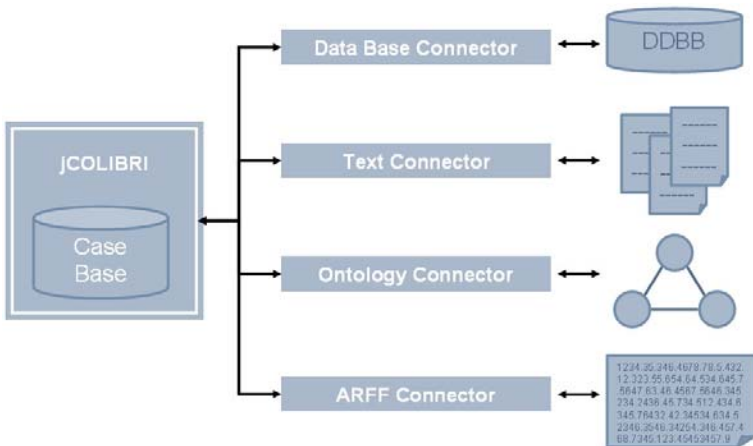


Fig. 4. Persistence organization in jCOLIBRI

connectors that work with plain text files, relational data bases and Description Logics (DL) systems. Other connectors can be included depending on the specific application requirements. This is the case of DI-CBR applications where large case bases must be loaded. To support a uniform way for loading these large case based, the *Thunder* extension provides a connector compatible with the ARFF format. This format is a standard defined by the popular WEKA toolkit for data mining and machine learning [17]. ARFF is currently a de-facto standard for large data bases and the inclusion of a connector in jCOLIBRI supporting this format supposes a great extension of the capabilities of the framework. Moreover, the storage of case bases using the WEKA format allows to compare results and techniques with the algorithms included in this toolkit: clustering, classification, etc.

### 3.2 In-Memory Organization

In the same way, connectors provide a common interface for accessing the persistence layer, once cases are loaded they are organized in-memory following another common interface. This way, the organization and indexation chosen for the Case Base will not affect the implementation of the methods. Apart from the common interface, certain organizations may support additional operations on the Case Base that specific methods can exploit. This is the case of the clustered organization because it provides methods to access the director vectors of each cluster.

The in-memory organization of the case base (linear, k-d trees, clusters, etc.) may have a big influence on the CBR processes, so the framework leaves open the election of the data structure that is going to be used. By default, jCOLIBRI provides four organizations: linear, cached linear (changes are not directly reflected in the persistence media), indexed by case id, and the new clustered organization.

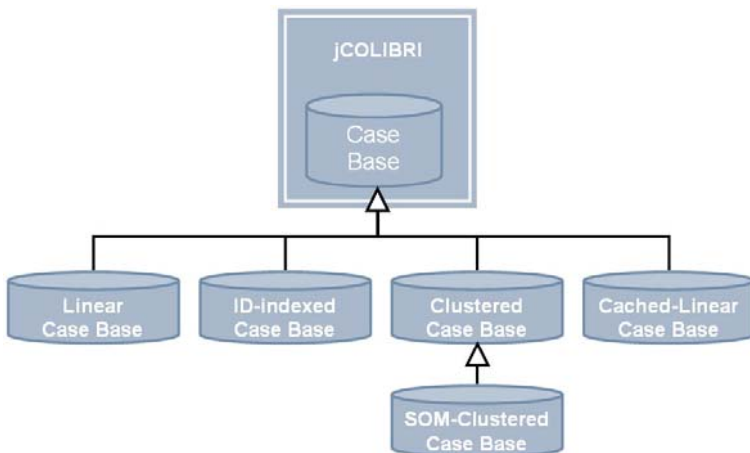


Fig. 5. In-memory organization of case bases in jCOLIBRI



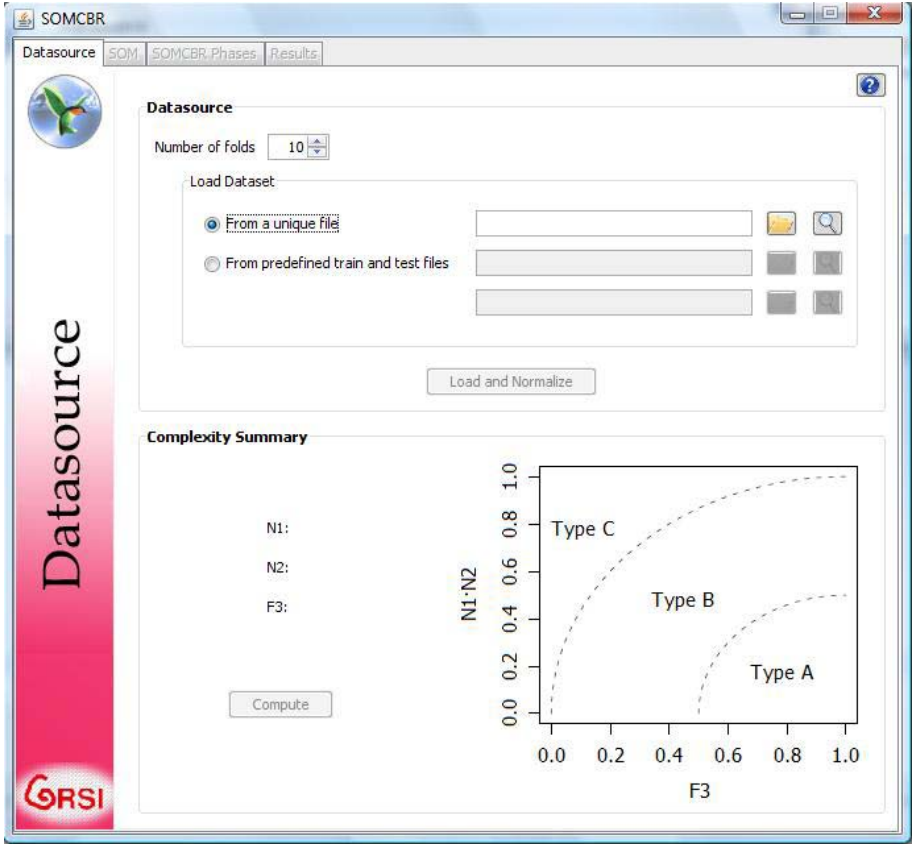
As is shown in Figure 5, there is a generic interface called *Clustered Case Base* that defines the common functionality that any clustered organization of cases must obey. This way, it makes possible to include any clustering techniques for indexing the case base. *Thunder* incorporates as default Self-Organizing Maps as clustering technique (see Section 4.2). The inclusion of clustering organizations for a case base must be performed taking into account the computation time required to organize the memory of cases. This process is usually very time consuming and may be configured with different similarity measures to group the cases. As detailed next, both requirements are smartly addressed in jCOLIBRI thanks to its organization of CBR applications and the definition of similarity functions:

**Reduction of computational time.** jCOLIBRI splits the CBR applications in precycle, cycle and postcycle. The precycle is executed only once when launching the application and is in charge of preprocessing the data and cases involved in the reasoning. This stage is very useful when managing textual cases because natural language processing tasks are very time consuming. However, the incorporation of clustered memories has shown that this stage is also very suitable because it frees the cycle for computing the clustering algorithms. This way, the *Thunder* plug-in performs the indexing into clusters of the case base in the precycle. Then, the cycle uses this pre-computed indexes to retrieve the cases in an efficient way. Finally, the postcycle is in charge of releasing the loaded resources.

**Computing similarities.** The other element involved in the clustering of the memory of cases is the similarity between cases. Similarity measures are applied to compute the similarity between two cases when computing the clusters and during the retrieval process. Of course, a correct retrieval of cases must use the same similarity function that was applied to generate the clusters in the case base. This requirement is easily solved through the simple interfaces defined in jCOLIBRI to implement similarity functions: compare two objects and return a value representing the similarity. This way, the simplicity of the interface allows to use these similarity functions both in the clustering and retrieval processes. Moreover, it implies that the large number of similarity functions already included in jCOLIBRI can also be reused for clustering the case base.

### 3.3 Execution of Case Retrieval Strategies

*Thunder* offers the possibility of applying the case retrieval strategies defined in the strategy map through a graphical interface or directly invoking the method at a source code level. The map configures a two-step retrieval method for clustered case bases. This new method retrieves cases by comparing initially the query with the prototype of each cluster and then comparing with the cases in the most similar clusters. As we have explained, the first step is to characterize the



**Fig. 6.** Screenshot of the graphical environment for loading the dataset and also for identifying its complexity. The complexity of the dataset used in the experiments (Section 4.3) is characterized as type B.

dataset as the screenshot of Figure 6 shows. In this case, the loaded dataset is the reported in Section 6 and its complexity is characterized as type B. Next, the case memory has to be organized in clusters applying the clustering algorithm selected. *Thunder* allows to load previously computed clusters or compute them in real time during the precycle step. Next, the desired improvement must be selected according to the scatter plots and the complexity of the dataset. These plots are obtained from the evaluation of many configurations over a wide set of datasets [18]. In this case, the scatter plots associated to SOM are available in [9]. Finally, the number of clusters and cases to use in the retrieval process is selected as the screenshot of Figure 7 shows. In addition, *Thunder* also includes others functionalities to promote the reuse phase [19] and the retain phase [20].

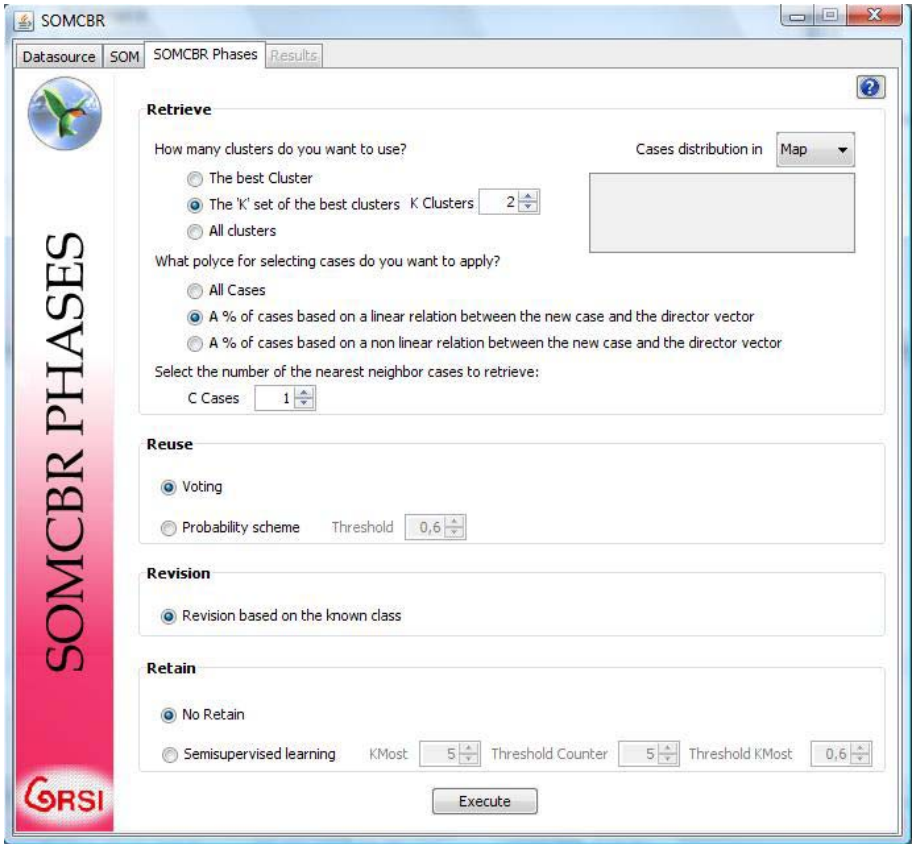


Fig. 7. Graphic tool for computing the different process of the CBR cycle

## 4 Case Study: SOM-Clustered Case Bases for Textual CBR

To illustrate the capabilities of the clustering techniques included in jCOLIBRI we have chosen the Textual CBR field. Although it is not very common to apply this kind of memory organizations with textual cases, we consider that this type of CBR applications could be greatly improved by applying clustered organizations of the texts. First, the textual CBR domain is presented. Next, a brief review of SOM is described. Finally, the results of analyzing this domain are discussed.

### 4.1 Textual CBR

Textual case-based reasoning (TCBR) is a subfield of CBR concerned with research and implementation on case-based reasoners where some or all of the knowledge sources are available in textual format. It aims to use these textual knowledge sources in an automated or semi-automated way for supporting problem solving through case comparison [21].

There does not appear to be a standard or consensus about the structure of a textual CBR system. This is mainly due to the different knowledge requirements in application domains. For classification applications only a basic stemmer algorithm and a cosine similarity function is typically needed, while with other applications more intense NLP derived structures are employed [22,23].

jCOLIBRI already includes methods for implementing both approaches for TCBR [14,24]. However, the management of large corpora of documents and the associated performance problems have been always an important drawback. An efficient management of large corpora is very important if we realize that experiences (cases) are described using natural language in many scenarios where CBR can be successfully applied. This way, CBR techniques must be able to manage efficiently a large amount of textual documents to leave the academic field and reach the commercial world.

## 4.2 Organization Based on Self-Organizing Maps

The improvement of performance through case memory organization has been tackled from many points of view, such as representing the attributes in tree structures [25] or graphs [26], grouping cases by their similarity [27], applying knowledge-intensive approaches [28] or data-intensive approaches [6]. Nevertheless, many of these methods pay more attention to how to structure the data rather to the underlying features of the domain: uncertainty and partial knowledge. In this scenario, *Soft-Computing* techniques are more suitable than *Hard-Computing* techniques since they are able to manage this kind of knowledge [29,30]. In particular, *Self-Organizing Map* (SOM) [8] has become one of the most used Soft-Computing clustering techniques to visualize and organize data [31] thanks to its capability for discovering hidden patterns.

SOM translates complex and nonlinear statistical relations contained in high-dimensional data into simple geometric relations on a low-dimensional space. The architecture of this neural network is constituted of two layers: (1) the input layer composed of  $N$  neurons, where each neuron represents one of the  $N$ -dimensional features of the input case; and (2) the output layer composed of  $M \times M$  neurons, where each one represents a set of similar cases by a director vector of  $N$  dimensions. Each input neuron is connected to all the output neurons. When a new input case  $C$  is introduced in the input layer, each neuron  $X$  from the output layer computes a degree of similarity between its director vector and the input case  $C$  applying a distance such as the normalized Euclidean distance. The result of the distance measures the level of cluster representativeness in relation to the input case. This smart and powerful ability is the reason why some CBR's studies focus on indexing the case memory for improving the retrieve phase from the point of view of clustering [32,33].

Although the *Thunder* contribution implements SOM for clustering the case base, it is important to note that any other algorithm could be easily applied because this possibility was taken into account when developing the contribution.

### 4.3 Experiments, Results and Discussion

Texts provide an ideal field for testing data intensive techniques that manage uncertainty knowledge. In our experiments, we chose a corpus of 1.500 documents. These documents were taken from several electronic journals and each one had an associated category: laws, history, medicine, and so on. These categories were useful to measure the performance of the retrieval process. Documents were processed to remove stop words and extract the stem. Then the TF-IDF filter was applied to select the most important 1.000 terms. The total number of categories was 20 and the precision was obtained through the leave-one-out testing method. Instead of using the cosine measure, we decided to apply another similarity function named Kullback-Liebler Divergence [34] that reported better results when comparing texts.

The goal of the experiments was to measure the improvement in the performance when using a linear case base and a clustered organization. To assign a category for a new query we used the majority-voting approach that selects the most repeated category of the  $k$  nearest neighbors. Although we tested several values for this  $k$  value, there were not significative differences in the results, so we will skip this parameter. The other parameter measured in the experiment was the number  $c$  of clusters used in the retrieval. Once the query was compared to each director vector (the prototype that represents each cluster), the  $c$  most similar clusters were used in the second step of the retrieval. Here, every case in these  $c$  clusters was compared to the query. This way, we were using the strategy number 4 in Figure 1.

Results (see Figure 8) illustrates how the precision changes according to the percentage of cases used in the retrieval. This percentage of cases depends on the number of clusters involved in the second stage of the retrieval process. Retrieving from all clusters means that the system is using a linear organization of the case base. As the number of clusters decreases, the percentage of cases compared to the query also decreases. Since our goal consists on measuring the impact in the precision when using just a part of the case base (some clusters),

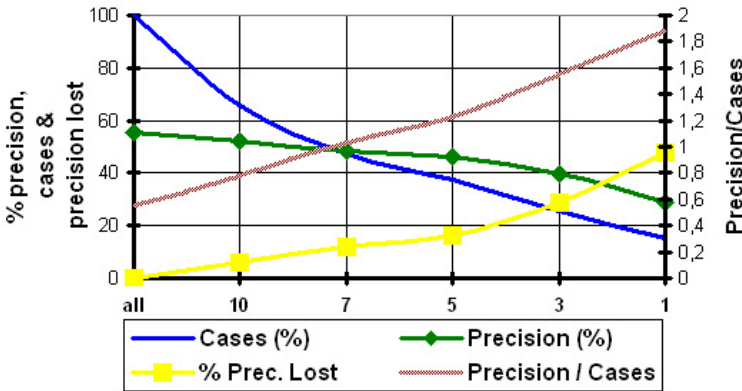


Fig. 8. Outline of experimental results

we also include in Figure 8 the value of the precision divided by the number of cases used in the retrieval. Finally, another point to be highlighted is the loss of precision with regard to the linear case base. Although the precision of the system may seem too low (independently of using a clustered or linear organization), it is important to note that the presented problem was very complex because of the high number of possible categories. Thus, a random classifier will obtain a 5% of precision having 20 different categories. So, precision values from 50% to 60% can be considered good results.

The experiment probes the improvement in the efficiency of the system when using a clustered memory of cases. For example, configuring a value of  $c=7$  (use the 7 most similar clusters), the precision only decreases a 10%, whereas the number of cases compared are half the size of the case base (47%). This important reduction of the cases involved in the retrieval implies a significant improvement in the efficiency of the CBR system.

## 5 Conclusions and Further Work

Over the last years open source tools for developing CBR applications has become a very interesting topic in the CBR community because it offers to the experts the possibility of testing and comparing their approach to others. However, there is not any open source framework designed for the management of clustered case memories. In this paper we have described *Thunder*, a plug-in that allows the integration of cluster-based retrieval processes in jCOLIBRI following the methodology defined in [9]. The main benefit of this approach is that computational time is reduced while the accuracy rate is maintained. To do it, a case retrieval based on two steps is followed: (1) a set of clusters represented by a prototype are selected and, (2) a set of cases from the clusters are used. *Thunder* offers the possibility of applying the case retrieval strategies defined in the strategy map through a graphic interface or directly invoking the method at a source code level. Moreover, *Thunder* incorporates a clustering algorithm called SOM for testing the tool and a connector for working with ARFF files.

We have tested the applicability of the tool in a textual case base where we have demonstrated how to increase the efficiency of retrieval through clusters to organize the case base. The experiment has shown the improvement in the retrieval performance when using a clustered organization of the textual case base. We have shown how the cluster-based strategy behaves differently according to the configuration of two parameters: the number of clusters selected and the number of cases retrieved from the selected clusters.

*Thunder* is available and it will be distributed with jCOLIBRI 2 next release. As future work, we are including other clustering techniques in *Thunder*, test the applicability of the clustering organization techniques in knowledge intensive ontology based domains and promoting the understanding of relation between data to promote the rest of CBR steps. Thus, it will be possible to perform a wide comparison among all the clustering strategies for organizing the case memory in order to find out the most suitable approach.

## Acknowledgments

We would like to thank the Spanish Government for the support in MID-CBR project under grant TIN2006-15140-C03 and the *Generalitat de Catalunya* for the support under grant 2005SGR-302. We would like to thank *Enginyeria i Arquitectura La Salle* of Ramon Llull University and *Universidad Complutense de Madrid* for the support to our research groups as well.

## References

1. Schulz, S.: CBR-works - a state-of-the-art shell for case-based application building. In: Procs. 7th German Workshop on CBR, GWCBR 1999, pp. 3–5. Springer, Heidelberg (1999)
2. Schumacher, J.: Empolis Orange – an open platform for knowledge management applications. In: 1st German Workshop on Experience Management (2002)
3. Stahl, A., Roth-Berghofer, T.: Rapid prototyping of CBR applications with the open source tool myCBR. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 615–629. Springer, Heidelberg (2008)
4. Bogaerts, S., Leake, D.: Iucbrf: A framework for rapid and modular case-based reasoning system development. Technical Report 617, Indiana University (2005)
5. Díaz-Agudo, B., González-Calero, P.A., Recio-García, J., Sánchez, A.: Building CBR systems with jcolibri. Special Issue on Experimental Software and Toolkits of the Journal Science of Computer Programming 69(1-3), 68–75 (2007)
6. Vernet, D., Golobardes, E.: An unsupervised learning approach for case-based classifier systems. Expert Update. The Specialist Group on Artificial Intelligence 6(2), 37–42 (2003)
7. Fornells, A., Golobardes, E., Vernet, D., Corral, G.: Unsupervised case memory organization: Analysing computational time and soft computing capabilities. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 241–255. Springer, Heidelberg (2006)
8. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Heidelberg (2000)
9. Fornells, A., Golobardes, E., Martorell, J., Garrell, J., Bernadó, E., Macià, N.: A methodology for analyzing the case retrieval from a clustered case memory. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 122–136. Springer, Heidelberg (2007)
10. Basu, M., Ho, T.: Data Complexity in Pattern Recognition. In: Advanced Information and Knowledge Processing. Springer, Heidelberg (2006)
11. Bernadó, E., Ho, T.: Domain of competence of XCS classifier system in complexity measurement space. IEEE Transaction Evolutionary Computation 9(1), 82–104 (2005)
12. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
13. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
14. Recio-García, J.A., Díaz-Agudo, B., Gómez-Martín, M.A., Wiratunga, N.: Extending jCOLIBRI for textual CBR. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 421–435. Springer, Heidelberg (2005)

15. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A., Sánchez-Ruiz-Granados, A.: Ontology based cbr with jcolibri. In: Applications and Innovations in Intelligent Systems XIV. In: SGAI 2006, pp. 149–162. Springer, Heidelberg (2006)
16. Díaz-Agudo, B., González-Calero, P.A.: An Ontological Approach to Develop Knowledge Intensive CBR Systems. In: Ontologies in the Context of Information Systems, pp. 173–213. Springer, Heidelberg (2007)
17. Witten, I., Frank, E.: Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco (2000)
18. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: jCOLIBRI 2 Tutorial. Technical Report IT/2007/02, Departamento de Ingeniería del Software e Inteligencia Artificial. Universidad Complutense de Madrid (2007), ISBN 978-84-691-6204-0, <http://gaia.fdi.ucm.es/projects/jcolibri/jcolibri2/docs.html>
19. Fornells, A., Golobardes, E., Martorell, J.M., Garrell, J.M.: Patterns out of cases using kohonen maps in breast cancer diagnosis. *International Journal of Neural Systems* 18, 33–43 (2008)
20. Fornells, A., Golobardes, E.: Case-base maintenance in an associative memory organized by a self-organizing map. In: Corchado, E., Corchado, J., Abraham, A. (eds.) *Innovations in Hybrid Intelligent Systems*, vol. 44, pp. 312–319. Springer, Heidelberg (2007)
21. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. *The Knowledge Engineering Review* 20(3), 255–260 (2006)
22. Brown, M., Förtsch, C., Wißmann, D.: Feature extraction - the bridge from case-based reasoning to information retrieval. In: *Proceedings of the 6th German Workshop on Case-Based Reasoning (GWCBR 1998)* (1998)
23. Brüninghaus, S., Ashley, K.D.: The role of information extraction for textual CBR. In: Aha, D.W., Watson, I. (eds.) *ICCBR 2001*. LNCS, vol. 2080, pp. 74–89. Springer, Heidelberg (2001)
24. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Textual CBR in jcolibri: From retrieval to reuse. In: Wilson, D.C., Khemani, D. (eds.) *Proceedings of the ICCBR 2007 Workshop on Textual Case-Based Reasoning: Beyond Retrieval*, August 2007, pp. 217–226 (2007)
25. Wess, S., Althoff, K., Derwand, G.: Using k-d trees to improve the retrieval step in case-based reasoning. In: Wess, S., Richter, M., Althoff, K.-D. (eds.) *EWCBR 1993*. LNCS, vol. 837, pp. 167–181. Springer, Heidelberg (1994)
26. Lenz, M., Burkhard, H., Brückner, S.: Applying case retrieval nets to diagnostic tasks in technical domains. In: Smith, I., Faltings, B.V. (eds.) *EWCBR 1996*. LNCS, vol. 1168, pp. 219–233. Springer, Heidelberg (1996)
27. Yang, Q., Wu, J.: Enhancing the effectiveness of interactive cas-based reasoning with clustering and decision forests. *Applied Intelligence* 14(1) (2001)
28. Rissland, E.L., Skalak, D.B., Friedman, M.: Case retrieval through multiple indexing and heuristic search. In: *IJCAI 1993*, pp. 902–908 (1993)
29. Cordon, O., Herrera, E.: Special issue on soft computing applications to intelligent information retrieval on the internet. *International Journal of Approximate Reasoning* 34, 2–3 (2003)
30. Cheetham, W., Shiu, S., Weber, R.: Soft case-based reasoning. *The Knowledge Engineering*, 1–4 (2005)



31. Oja, M., Kaski, S., Kohonen, T.: Bibliography of Self-Organizing Map (SOM) Papers: 1998-2001 (2003), <http://www.cis.hut.fi/research/refs/>
32. Chang, P., Lai, C.: A hybrid system combining self-organizing maps with case-based reasoning in wholesaler's new-release book forecasting. *Expert Syst. Appl.* 29(1), 183–192 (2005)
33. Fornells, A., Armengol, E., Golobardes, E.: Retrieval based on self-explicative memories. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 210–224. Springer, Heidelberg (2008)
34. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2008)

# Case-Based Collective Inference for Maritime Object Classification

Kalyan Moy Gupta<sup>1</sup>, David W. Aha<sup>2</sup>, and Philip Moore<sup>1</sup>

<sup>1</sup> Knexus Research Corporation; Springfield, VA 22153

<sup>2</sup> Navy Center for Applied Research in Artificial Intelligence,  
Naval Research Laboratory (Code 5514), Washington, DC 20375

firstname.lastname@kexusresearch.com, david.aha@nrl.navy.mil

**Abstract.** Maritime assets such as merchant and navy ships, ports, and harbors, are targets of terrorist attacks as evidenced by the USS Cole bombing. Conventional methods of securing maritime assets to prevent attacks are manually intensive and error prone. To address this shortcoming, we are developing a decision support system that shall alert security personnel to potential attacks by automatically processing maritime surveillance video. An initial task that we must address is to accurately classify maritime objects from video data, which is our focus in this paper. Object classification from video images can be problematic due to noisy outputs from image processing. We approach this problem with a novel technique that exploits maritime domain characteristics and formulates it as a graph of spatially related objects. We then apply a case-based *collective classification* algorithm on the graph to classify objects. We evaluate our approach on river traffic video data that we have processed. We found that our approach significantly increases classification accuracy in comparison with a conventional (i.e., non-relational) alternative.

## 1 Introduction

Maritime assets such as merchant and naval vessels are under a constant threat of terrorist attacks. For example, the USS Cole was completely disabled in a bombing event at the Port of Yemen that claimed the lives of 11 sailors.<sup>1</sup> Existing approaches to counter such threats use a combination of sensors such as radar, video surveillance, and manual *watchstanding*. These approaches are manually intensive; potential threats can be overlooked due to human factors such as information overload and fatigue. In addition, sensors such as radar are largely ineffective against small, fast moving vessels. We are developing a decision support system, named the *Maritime Activity Analysis Workbench* (MAAW), to address some of these problems. MAAW is being designed to detect potentially threatening surface vessels by automatically processing maritime surveillance video. A critical task in this context is that of classifying maritime objects. If accurately predicted, MAAW can then assess their potential threat and issue alerts to *watchstanders*, our primary end users.

---

<sup>1</sup> [http://en.wikipedia.org/wiki/USS\\_Cole\\_bombing](http://en.wikipedia.org/wiki/USS_Cole_bombing)

Our object classifier must operate on potentially noisy data obtained from image processing components yet perform robustly. When combined with a large number of closely related vessel types, this poses a significant challenge for conventional classification methods, which classify objects independently. However, in the maritime domain, we expect the context of maritime objects to provide important clues for object classification. For example, a tugboat in a harbor often tows a cargo vessel. Given that images of small vessels are harder to accurately classify than those of large cargo vessels, their proximity could be an important clue for classifying tugboats. One approach for incorporating contextual clues is *collective classification*. Unlike conventional approaches, collective classifiers *concurrently* classify a set of *related* objects. This approach has not previously been applied to the task of maritime object classification, and its effectiveness on this task is unknown.

Our focus in this paper is an object classification method that exploits contextual cues in a maritime video scene. Our contributions are as follows. First, we frame our task as that of using contextual relational cues to increase object classification accuracy. We represent these cues by transforming a maritime scene into a graph of spatially related objects. We then apply a case-based collective classifier, which includes a conventional classifier, domain-specific parameterized similarity measures (learned from the data), and a collective inference procedure. Finally, we evaluate our approach on maritime river traffic video captured by our system. We found that case-based collective classification significantly outperforms a conventional independent object classification approach on this task.

We briefly describe the topic of maritime surveillance and related work in Section 2. In Section 3, we present an overview of our system. In Section 4, we describe its algorithm for case-based collective classification of maritime objects. We discuss the evaluation of our approach in Section 5. Finally, we conclude the paper with remarks on future research.

## 2 Maritime Video Surveillance and Related Work

Other researchers have addressed maritime domain awareness tasks not unlike the port/harbor security task that is our ultimate focus. For example, Rhodes et al. (2005) employ neural network classifiers (specifically, a modification of Fuzzy ARTMAP) to learn normalcy models for anomaly detection in maritime environments. As with MAAW, their objective is for operators to provide feedback for learning. We are addressing the task of using spatial and temporal relations to detect coordinated activities. However, their data combines metadata with automated identification systems (AIS) data, rather than video imagery.

ObjectVideo<sup>2</sup> deploys sophisticated products for maritime (and other types of) intelligence surveillance, including those for real-time, high-speed, activity-based video indexing and retrieval. This can be used, for example, to perform forensic analysis (e.g., detect the movement of suspicious objects). Motivated by the fact that humans cannot monitor a vast number of vessels/objects simultaneously, their system automatically extracts object descriptions (from a variety of sensors) using a statistical pixel modeling approach, and employs user-provided rules to determine when to generate security alerts (Lipton *et al.*,

---

<sup>2</sup> <http://www.objectvideo.com>

2009). A key difference of our approach is that we instead use a case-based statistical relational learning approach for object recognition.

There is a long and rich history of research in case-based reasoning (CBR) on image processing (Perner *et al.*, 2005). Most of these have focused on images rather than video. For example, Perner's (1993) cases are graphically represented using spatial relations, and structural similarity is computed to classify weld defects. In contrast, our cases are *related* spatially and are not represented using only intrinsic attributes. Also, we instead use collective inference for object classification.

Work on video data within CBR has primarily concerned methods for video retrieval. For example, Burke and Kass (1995) describe an approach for retrieving and presenting stories from a video data base to support educational goals. Similarly, Johnson *et al.* (2000) describe an ASK system (the Air Campaign Planning Advisor) in which educational video recordings of domain experts can be retrieved through a tailored interface and a hierarchical task decomposition model. Zhang and Nunamaker (2004) index videos by their metadata. Their system retrieves cases using natural language queries. While this genre of research focuses on user interfaces and video retrieval, our focus in this paper is instead on object recognition in video data.

Some other uses of video have been the focus of CBR and related research. For example, MacNeil (1991) describes a visual programming environment in which CBR is used to capture and reuse knowledge pertaining to the creation of multimedia presentations. In MacNeil's TYRO, cases are generic temporal templates, abstracted from video segments, which denote chunks of design experience. These general cases can then be used to provide constraints for creating similar videos. In contrast, we focus on specific case representations, and recognizing objects from video.

Our ultimate focus is on threat analysis: if we can accurately identify the objects in these videos and recognize their behaviors, our next step will be to assess whether these behaviors are threatening (i.e., to naval/maritime assets). A variety of CBR research has focused on threat analysis techniques. For example, Jung *et al.* (1999) use CBR to perform risk analysis for organizations in electronic commerce environments, where cases are used to evaluate assets and assess threats and vulnerabilities by comparison with prior security accidents. Multiple groups have also used CBR to assist intelligence analysts with constructing (Adams & Goel, 2007) and/or analyzing (Murdock *et al.*, 2003) hypothesized terrorist threats. CBR has also been used, several times, to detect anomalies in computer networks (e.g., see a recent investigation described by Micarelli and Sansonetti (2007)). A primary distinction of our work from these investigations is that we are working with video data.

Finally, a distinguishing feature of our approach is that we use collective classification to leverage the spatial and temporal relations among objects to increase predictive accuracy. Previous work on collective classification, a form of statistical relational learning (SRL), has not been applied to tasks involving video data (Sen *et al.*, 2008). This includes our own previous research on case-based collective classification (McDowell *et al.*, 2007a). However, other SRL approaches have been applied to similar tasks. In particular, Aboutalib and Veloso (2007) leverage human-provided cues, detected from humans interacting with objects in video data, to recognize those objects using probabilistic relational models. Unlike our work, they do not use a CBR approach for this task, nor focus on maritime object recognition.

### 3 The Maritime Activity Analysis Workbench

Our goal is to develop a decision support system for maritime security personnel (e.g., watchstanders, harbor masters) to assist them with their surveillance and decision making tasks and improve their threat assessment capability. To meet this goal, we are developing MAAW. It includes a series of adaptive processors, ranging from video acquisition to threat analysis, designed to interact with its user to issue alerts, provide threat assessments, and receive performance feedback with corrections (see Figure 1). A crucial component in its pipeline of processors is the Object Classifier, which we discuss in detail in Section 4. Here, we briefly review MAAW's intended functionality and explain the role of its Object Classifier.

MAAW's *Video Acquisition* subsystem currently includes a fixed video camera that captures maritime traffic video in black and white, digital format. It can also capture videos from online harbor cams. The acquisition system suitably compresses the video and hands it off to the *Video Processor* for further processing. The *Detector* within the Video Processor performs basic operations such as adaptive background subtraction to detect moving maritime objects such as boats and ships. The *Tracker*, also a component of the Video Processor, then groups the objects detected from a series of video frames into tracks. It uses a combination of *Appearance Models* (not shown in Figure 1) and clustering techniques to perform its task.

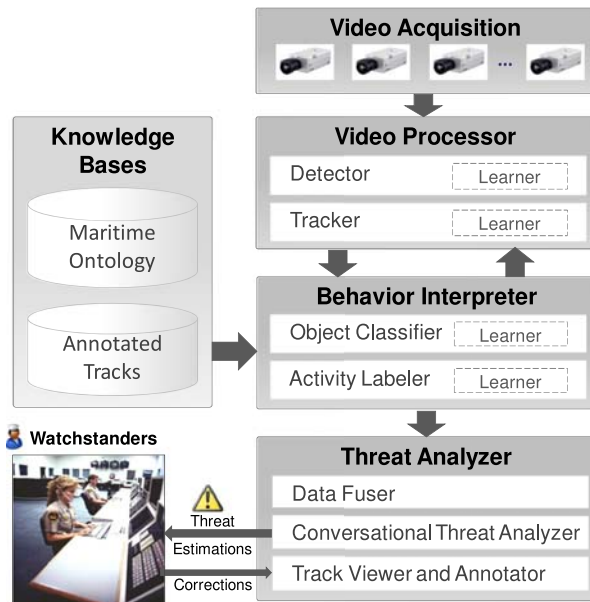


Fig. 1. MAAW's Functional Architecture

The Video Processor outputs track information in a data structure suitable for consumption by the *Behavior Interpreter*. The track is represented as a series of events, each referring to a maritime object and its attributes such as position, speed, and

image signature. The Behavior Interpreter's function is to classify the objects within a track and the activities they are performing. We are combining supervised learning approaches along with maritime surveillance domain knowledge to accomplish these tasks. The *Object Classifier* and the *Activity Labeler* are the two components in the Behavior Interpreter. They rely on two knowledge bases: a *Maritime Ontology*, and a database of *Annotated Tracks*. We are annotating all the track data with objects categories and activity descriptions chosen from this ontology, and expect users and subject matter experts to do so minimally while using MAAW.

The maritime object classification task can be challenging because tracks extracted by the Video Processor can be noisy depending on a variety of application conditions such as the weather, time of day, the size and the number of objects, and occlusion. For example, a single object in the scene could result in multiple spurious tracks with inaccurate attribute value estimates. In this paper, we explore one way to address this problem. We investigate whether taking application and scene context into account can increase classification accuracy, even when the track data is noisy.

The Behavior Interpreter hands off the automatically labeled tracks to the *Threat Analyzer*, which will fuse the labeled tracks with harbor database information to assess threats and issue alerts. End users will be able to accept or reject MAAW's decisions and provide corrective feedback, which MAAW will use to update the track database.

MAAW is, in part, an annotation tool for video processing. Rincón and Martínez-Cantos (2007) survey other such tools, which differ from MAAW in that they do not employ case-based collective classification to perform maritime object classification.

## 4 Case-Based Collective Classification of Maritime Objects

### 4.1 Collective Classification

Conventional classification methods assume that cases/instances are independently drawn from an identical distribution (the i.i.d. assumption), and are otherwise not related. However, in many practical classification tasks cases can be explicitly or implicitly related. For example, web pages can be explicitly related by hyperlinks and emails can be implicitly related as requests and responses. Likewise, in a maritime object classification task, cases that represent objects can be implicitly related spatially and/or temporally. For example, a tugboat track can be spatially related to a cargo-ship track that it is towing out of a harbor.

Relations across cases can provide valuable contextual information that can potentially be leveraged to increase classification accuracy. For example, *collective classifiers* use these relations to concurrently classify cases, which can often increase classification accuracy (Sen *et al.*, 2008). The magnitude of accuracy increase depends on a number of factors that characterize the related cases. In particular, accuracy is an increasing function of their autocorrelation (i.e., the correlation among attributes, and in particular the class labels, of related cases), a decreasing function of relation density, and a decreasing function of attribute predictiveness (i.e., their correlation with class label) (Jensen & Neville, 2002; Sen *et al.*, 2008). As explained below, the task of object classification on cases extracted from maritime video exhibits some of these characteristics. Therefore, it is a suitable candidate for collective classification.

Two broad categories of collective classification algorithms have been studied, as distinguished by how they perform inference:

1. *Local collective inference*: These algorithms operate on a vector space representation of attributes obtained by transforming a graph of related cases. The Iterative Classification Algorithm (ICA) (Neville & Jensen, 2000), Gradual Commit (McDowell *et al.*, 2007b), and Gibbs sampling (Geman & Geman, 1984) are some example techniques.
2. *Global collective inference*: These algorithms operate directly on a graph of related cases rather than attribute vectors. Examples in this category include loopy belief propagation (Pearl, 1988) and relaxation labeling (Rosenfeld *et al.*, 1976).

In this paper, we apply ICA to our task. We selected it due to its simplicity, efficiency, and comparatively good performance (Sen *et al.*, 2008).

Local collective classification algorithms operate on a representation of cases that includes both *intrinsic* and *relational* attributes, where the former describe properties of an individual case and the latter denote relations among cases. In this context, collective classification is a two-stage supervised learning process:

1. *Bootstrap classification*: Effective relational representations for a case typically include attributes defined as relations on the values of the class labels of related cases. For example, in the maritime domain, a relational attribute may include the distance of one track to another and the category label of the related object. However, the dependency of relational attribute values on class labels of related cases poses a problem: at the start of the classification, the class labels of the related cases are sometimes unknown, which implies that their relational attribute values cannot be computed. To jump start this process, an initial prediction for the class labels must be obtained. This is accomplished by applying a classifier to only the *intrinsic* attributes. Any conventional supervised learner (e.g., Naïve Bayes, SVMs, a case-based classifier) can be used for bootstrap classification. In this paper we use a simple case-based classifier.
2. *Collective inference*: This inherently parallel process is simulated by iterating over a loop of two steps:
  - a. *Predict relational attribute values*: Based on the class labels obtained in the previous step, these values are computed to complete the case representation.
  - b. *Perform local classification*: The classifier learned during the bootstrap step is used to classify cases with their predicted relational attribute values.

Typically, the accuracy of relational attribute value predictions and local classifications increase over subsequent iterations. For the ICA algorithm, iterations of collective inference cease when there are no changes to classification predictions in successive iterations, or after a predetermined max number of iterations. Empirical evaluations of ICA show that it typically converges in a relatively small number of iterations (e.g., 10) (McDowell *et al.*, 2007a).

In summary, the supervised classifier learned during the bootstrap step has access to only the (non-relational) intrinsic attributes, whereas it also has access to the relational attributes during collective inference.

In this paper, we assume no links between the training and test sets; this is known as the *out-of-sample* task (Neville & Jensen, 2005). Thus, the classifier is trained on a set of completely defined cases in step 2.b because the labels of related cases, from which the relational attribute values are derived, are all available (i.e., either given or predicted). We also assume that the relations to be used are pre-selected rather than learned. We address the implications of these and other assumptions in our evaluation in Section 5 and in the subsequent discussion in Section 6.

---

```

ICA (Tr,Te,NR,R,n,S)=
//Tr = Training data, Te = Test data, NR = non-relational features,
//R = rel.features, n = #iterations, S = supervised learner
1   Tr.R.values ←setRelFeatures(Tr,R)           //Relational value estimation
2   M←learnModel(Tr,NR,R,S)                     //Learn initial relation model
3   Te.Labels ←classify(Te,Tr,M,NR,∅)           //Bootstrap classification
4   for j=0 to n                                //Collective inference
5     Te.R.values ←setRelFeatures(Te,R)         //Relational value estimation
6     Te.Labels ←classify(Te,Tr,M,NR,R)        //Local classification
7   Return Te.Labels                            //Return final labels

```

---

Fig. 2. Pseudocode for the Iterative Collective Algorithm (ICA)

## 4.2 Case-Based Collective Inference

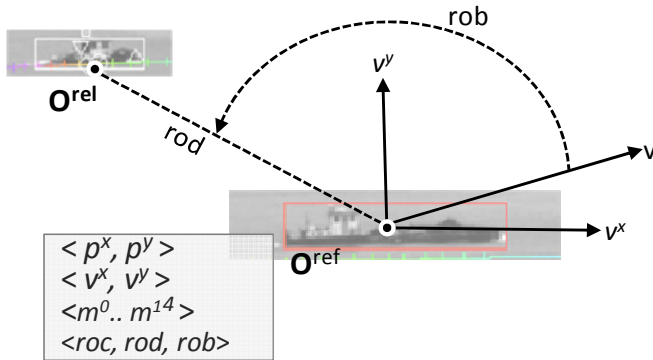
In Section 4.1, we described a simple collective classifier called the Iterative Classification Algorithm. Figure 2 presents ICA's pseudocode where, in this study, we use a case-based algorithm (for  $S$ ) to perform supervised learning and prediction.

Case-based classification predominantly involves retrieving similar cases and reusing their class labels to predict the label for a new classification problem (López de Mántaras *et al.*, 2005). Below we describe our case representation for maritime object classification, followed by the retrieval and reuse methods we use. Case retention can be important in an application like ours, but we leave it for future work.

**Case Representation:** The object classifier receives a structured representation of tracks as its input. A track comprises multiple events, each resulting from a change in an object's direction or speed. Ideally, a track represents a single moving maritime object. However, MAAW's Video Processor can make mistakes while grouping multiple events from a scene into multiple tracks. Our goal is to use the Object Classifier to reduce errors in categorizing images and use vessel category labels provided by the Object Classifier to correctly rebuild the tracks. That is, we classify objects for each event in a track instead of the track as a whole. Moreover, in our application, the track must be repeatedly classified as soon as it is detected and its classification revisited as the track unfolds. Therefore, *we represent each event in a track as a case within MAAW.*

We use a typical <problem, label> representation for our cases. Problems are represented by intrinsic and relational attributes. *Intrinsic* attributes of a case are those attributes of a maritime object that are independent of other objects. For our task, these include the following three groups of 19 attributes (see Figure 3):





**Fig. 3.** Attributes representing problems in cases denoting related maritime objects

1. *Object position*: This represents the position of a maritime object in a two-dimensional coordinate system detected and extracted by the Video Processor from the maritime video. It is a tuple  $\langle p^x, p^y \rangle$  comprising two continuous real values.
2. *Object velocity*: This represents the velocity vector (i.e., speed and direction) of a maritime object. Like object position, the velocity vector is represented in two dimensions using a tuple  $\langle v^x, v^y \rangle$  comprising two continuous real values.
3. *Object image moments*: Our Video Processor extracts images of objects from a scene including its shape, which it converts into a characteristic shape signature. Shape signatures or moments are a commonly used technique for analysis and comparison of 2D shapes. They capture information such as orientation, size, and shape boundary (Leu, 1991). We generate fourth order moments, which is a tuple comprising 15 real continuous values  $\langle m^0 \dots m^{14} \rangle$ .

In addition to these attributes, we employ the following group of *relational* attributes:

4. *Closest track object*: These three attributes encode the spatial relationship of a reference object (i.e., the object that the case represents) in a maritime scene to a related maritime object that is the closest to it. The distance between a reference object and a related object is computed based on their positions in the two-dimensional real world coordinates. The attributes comprise a tuple of three values  $\langle roc, rod, rob \rangle$ :
  - a. Related object category (*roc*): This is a categorical label of the related object selected from our Maritime Ontology.
  - b. Related object distance (*rod*): This is the distance of the related object from the reference object represented by a continuous real value greater than or equal to 0. (We define our distance function below.)
  - c. Related object bearing (*rob*): This is the angle between the velocity vector of the reference object and the position vector of a related object.

Other (e.g., temporal) relationships among objects exist that we could use for maritime object classification, but we leave their consideration for future study.

**Case Retrieval:** A new problem in MAAW refers to an unlabeled object in a maritime scene. We retrieve the  $k$  most similar stored cases by comparing each of them with the new problem to assess their overall similarity. We compute the overall similarity by a weighted aggregation of attribute similarities, where the definitions for each of the four groups of attributes are defined as shown below.

- i. *Positional similarity:* We compute the positional similarity  $PosSim(o_i, o_j)$  of two objects  $o_i$  and  $o_j$  as follows:

$$PosSim(o_i, o_j) = 1 - \text{dist}(o_i, o_j) / \text{MaxDist}, i \neq j \quad (1)$$

$$\text{dist}(o_i, o_j) = \sqrt{(p_i^x - p_j^x)^2 + (p_i^y - p_j^y)^2}$$

$$\text{MaxDist} = \sqrt{(\max(p^x) - \min(p^x))^2 + (\max(p^y) - \min(p^y))^2}$$

where  $\text{dist}()$  is the Euclidean distance between two maritime objects computed using the attributes representing their respective positions (i.e., the tuple  $(p^x, p^y)$ ).  $\text{MaxDist}$  is a similarity metric parameter representing the maximum possible distance for a pair of objects, computed from their position values over the entire case base.

- ii. *Velocity similarity:* We compute the similarity  $SpeedSim(o_i, o_j)$  of the speed of two maritime objects  $o_i$  and  $o_j$  as follows (we ignore directional differences for this task):

$$SpeedSim(o_i, o_j) = 1 - \text{diff}(o_i, o_j) / \sigma_s, i \neq j \quad (2)$$

$$\text{diff}(o_i, o_j) = \sqrt{(v_i^x - v_j^x)^2 + (v_i^y - v_j^y)^2}$$

where  $\sigma_s$  is a similarity metric parameter representing the standard deviation (i.e., variance) of object speeds over the entire case base.

- iii. *Moment similarity:* We compute the similarity  $MomSim(o_i, o_j)$  of the image moments of two maritime objects  $o_i$  and  $o_j$  as follows:

$$MomSim(o_i, o_j) = \sum mvSim^k(o_i, o_j) / 15, 0 \leq k \leq 14 \quad (3)$$

$$mvSim^k(o_i, o_j) = 1 - \min(l, |(mv_i^k - mv_j^k) / \sigma_m^k|) \quad (3.1)$$

Equation 3 averages the similarities across 15 moment value similarities, where each moment value similarity  $mvSim^k(o_i, o_j)$  is calculated using Equation 3.1, which computes the minimum of the proportional difference of the  $k^{\text{th}}$  moment values  $mv^k$ . This metric uses 15 parameters,  $\sigma_m^k$ , each representing the variance of the  $k^{\text{th}}$  moment value across the entire case base.

- iv. *Closest object similarity:* This metric assesses the similarity of pairs of spatially related objects. The attribute *closest track object* captures the spatial relation between a reference object and its closest related object. This metric,  $ClobSim(o_i,$

$o_j$ ), compares this relation in two parts (see Equation 4). First, it checks to see if the categories of related objects (i.e.,  $roc$ ) are the same. Then it compares the distance (i.e.,  $rod$ ) using  $rdistSim()$  and the bearing (i.e.,  $rob$ )  $rbearingSim()$ . Equation 4 averages the distance and bearing similarities.

The distance similarity is computed using Equation 4.1, which uses a metric parameter,  $\sigma_{rod}$  which represents the variation of  $rod$  across the entire case base. The bearing similarity is computed in four parts based on the four quadrants of a circle centered on the reference object ( $\delta/2, \delta, 3\delta/2, 2\delta$ ) that roughly represent the forward, rightward, backward, and leftward topological spaces of an object. These are forward similarity ( $fsim$ ), backward similarity ( $bsim$ ), rightward similarity ( $rsim$ ) and leftward similarity ( $lsim$ ), respectively:

$$ClobSim(o_i, o_j) = 0, \text{if } (roc_i \vee roc_j = "NONE") \vee (roc_i \neq roc_j) \quad (4)$$

$$= (rdistSim(o_i, o_j) + rbearingSim(o_i, o_j)) / 2, \text{otherwise} \quad (4.1)$$

$$rdistSim(o_i, o_j) = 1 - |rod_i - rod_j| / \sigma_{rod}$$

$$rbearingSim(o_i, o_j) = \min(\min(fsim, bsim), \min(rsim, lsim))$$

$$\text{for } \theta = rob_i - rob_j$$

$$fsim(\theta) = 2 * |\pi/2 - \theta| / \pi \quad \text{when } 0 < \theta < \pi/2$$

$$= 1 - 2 * |2\pi - \theta| / \pi \quad \text{when } 3\pi/2 < \theta < 2\pi$$

$$= 0 \quad \text{otherwise}$$

$$bsim(\theta) = 1 - 2 * |\pi/2 - \theta| / \pi \quad \text{when } \pi < \theta < 3\pi/2$$

$$= 1 \quad \text{when } \theta = \pi$$

$$= 0 \quad \text{otherwise}$$

$$rsim(\theta) = 1 - 2 * |3\pi/2 - \theta| / \pi \quad \text{when } \pi < \theta < 2\pi$$

$$= 1 \quad \text{when } \theta = 3\pi/2$$

$$= 0 \quad \text{otherwise}$$

$$lsim(\theta) = 1 - 2 * |\pi/2 - \theta| / \pi \quad \text{when } 0 < \theta < \pi/2$$

$$= 1 \quad \text{when } \theta = \pi/2$$

$$= 0 \quad \text{otherwise}$$

The function we use to compute aggregate similarity  $Osim(o_i, o_j)$  for the learned classifier is as follows:

$$Osim(o_i, o_j) = (PosSim(o_i, o_j) + SpeedSim(o_i, o_j) + MomSim(o_i, o_j)) / 3 \quad (5)$$

$$Osim(o_i, o_j) = (PosSim(o_i, o_j) + SpeedSim(o_i, o_j) + MomSim(o_i, o_j) + ClobSim(o_i, o_j)) / 4 \quad (6)$$

where Equation 5 refers to the computation before relational values have been computed (i.e., during the bootstrap phase) and Equation 6 refers to the situation after the relational values have been computed (i.e., during collective inference). For the sake of simplicity, we ignore differential weighting of features in this paper, leaving this for future study.

**Case Reuse:** We use the similarity-weighted voting kernel function for reusing the labels from the  $k$  most closely matching cases. This kernel collates the votes for the candidate category labels from each of the  $k$  cases, where each offers its  $Osim()$  value

as a vote toward its object label. The kernel then computes the total vote for each candidate label by summing over all the votes it receives, and selects the label with the largest vote as the label for the new problem.

**Supervised Learning:** Learning a case-based classifier can include learning/tuning its similarity metric from a memory of stored cases. This can involve, for example, feature weight learning and computing the values of metric parameters. In this paper, we perform only this latter task. We computed the settings of the parameters for each of the four parameters described above (i.e., MaxDist for positional similarity,  $\sigma_s$  for velocity similarity,  $\sigma_m^k$  for moment similarity, and  $\sigma_{rod}$  for closest object similarity). This entails estimating their value over the entire case base. For example,  $\sigma_{rod}$  is the standard deviation, a statistic computed over the real-valued attribute *rod*.

## 5 Evaluation

### 5.1 Objective

Our objective was to evaluate whether using a collective classification approach for our maritime object classification task attains a significantly higher accuracy than does a conventional supervised learning algorithm. In other words, we formulate the following null hypothesis:

**H<sup>0</sup>** There is no difference between the maritime object classification accuracy obtained by a collective classifier and the accuracy obtained by a conventional but otherwise equivalent supervised learning algorithm.

### 5.2 Method

**Data:** We selected two days of video of maritime activities on the Potomac River in Washington, DC. We used the Video Processor on this video to detect tracks of moving maritime objects and their attributes (e.g., position and velocities at different points in time). Using MAAW, we then labeled all the events in a track with appropriate object categories (see Figure 4). These object category labels were chosen from a Maritime Ontology (a taxonomy of objects, partially visible in Figure 4) that we developed using MAAW. Typically, leaf nodes of the ontology were selected, but subject matter experts were also allowed to select intermediate nodes when the object could not be visually categorized at the most specific level.

Our database included 1578 cases of labeled objects. The database included cases in 23 object categories from our Maritime Ontology, with proportions ranging from 46.64% to 0.13%. The top three most populous labels were *wave* (46.64%), *small-touring-vessel* (9.76%), and *wake* (7.41%). Half the object categories (e.g., *steam-paddle-touring-vessel*) were relatively rare and occurred less than 2% of the time in our data set.

**Algorithms:** We implemented two algorithms to conduct a comparative empirical evaluation. They were implemented using the Knexus Classification Workbench (KCLAW), a proprietary Java library for classification tasks:

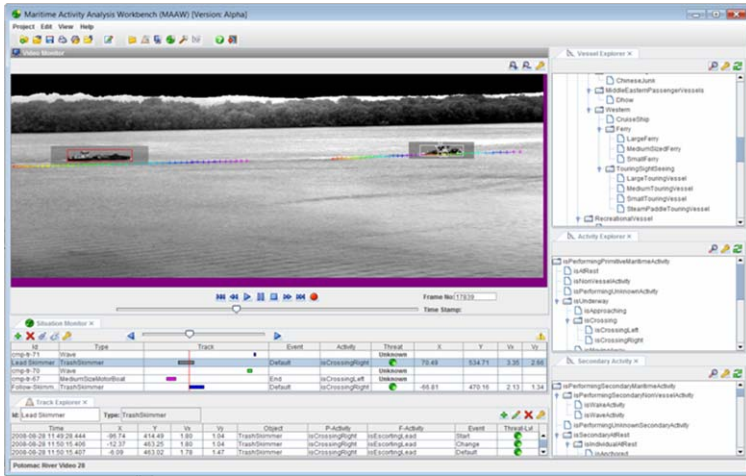


Fig. 4. MAAW can be used to label the extracted maritime tracks

1.  $ICA_0$ : This is a conventional case-based classifier (kNN) that does not perform collective classification. It differs from ICA in that it performs no collective inference, and does not employ relational attributes.
2. ICA: We summarized this simple collective classifier in Section 4.1 and detailed its application to our maritime object classification task in Section 4.2.

**Performance Measure:** We used classification accuracy as the performance measure with some modification. Given the nature of our domain, we considered graded misclassification costs based on the Maritime Ontology of object labels. In particular, we permit misclassification costs to be less than 1, depending on the taxonomic relationship between the correct and predicted labels. To do this, we used the Maritime Ontology to compute a misclassification cost matrix. For example, if a *small-motorboat* was classified as a *medium-sized-motorboat* the classification error was 0.5 rather than 1.0 because they are siblings in this taxonomy.

**Test Procedure:** We adopted a leave-one-out cross validation (LOOCV) test procedure with some modifications. Conventional LOOCV procedures use one case from the database for testing and the remainder for training, cycling through the entire case base and averaging the results of individual tests. We cannot use this here because collective inference operates on a *graph* of related cases, and we choose to eliminate any relations between the training and test cases. Therefore, we grouped cases that refer to co-occurring tracks and events within the same track; each such grouping yields a single fold (i.e., each fold's cases have no relations with cases in other folds). Next, we treated each fold as a test set and the union of cases from the remaining folds as the corresponding training set (i.e., the case base). This yields 177 folds, of which 77 contain cases with relational attribute values. The average number of cases-per fold across the entire data set is 8.92. The average number of cases in relational folds was marginally greater (10.79).  $ICA_0$  and ICA were applied to each test set (i.e., fold) and their classification accuracy was recorded.

**Analysis:** We used a paired student's 1-tailed t-test to evaluate the null hypothesis  $H^0$ .

**Table 1.** Average Classification Accuracies for the Collective and Non-Collective Classifiers on the Maritime Object Classification Task

Comparison Scope	ICA <sub>0</sub>	ICA	Significance
Relational Only	46.90	51.85	0.0001
Overall	53.23	56.06	0.0019

### 5.3 Results and Analysis

We compared the performance of ICA<sub>0</sub> and ICA under two dataset conditions:

1. *Relational Only*: To obtain insight on their true performance differences, we compared the algorithms using *only* those 77 folds that contain relations.
2. *Overall*: To assess the overall impact of collective inference (at least, as embodied in ICA) for our application, we compared the two algorithms using *all* 177 folds to obtain an aggregate performance measure.

Table 1 summarizes the results. The average classification accuracies of ICA<sub>0</sub> and ICA for the Relational Only condition are 46.90% and 51.85% respectively ( $p=0.0001$ ). Thus, we reject our null hypothesis  $H^0$  and confirm that ICA, a case-based collective classifier, attains significantly higher accuracy than does an otherwise equivalent conventional (i.e., non-relational, non-collective) case-based classifier for our maritime object classification task. For the Overall condition, ICA still significantly outperforms ICA<sub>0</sub> (i.e., 53.23% and 56.06% respectively ( $p=0.0019$ )) although, as expected, their performance difference is smaller (4.95 vs. 2.83). There are a large number of classes in our domain and many of them occur rarely. Thus, ICA<sub>0</sub>'s classification accuracy for the Relational Only condition is substantially lower than it is for the Overall condition.

## 6 Discussion

Our algorithm benefited greatly from experimenting with alternative similarity functions. For example, while not reported here, we found no benefit for the collective classifier until we used a similarity metric that transformed the bearing into topological quadrants. Although we compared the performance of our algorithms using a graded (non-binary) classification error measure, our conclusions remain valid when we use a binary classification measure.

Performance could be further improved by using higher quality data and refining the collective classification algorithm. First, the data we are using is noisy; there are large variations in position detection (e.g., the position at which an object is detected can be inaccurate due to low-resolution imagery). Also, the shape geometry uses coarse techniques. We are currently addressing these issues. Also, we plan to improve tracking by providing feedback from the Behavior Interpreter to the Video Processor (see Figure 1) so as to facilitate the learning of more accurate appearance models.

Second, ICA's behavior could be improved. While we are using the *closest track object* relation, we have not yet examined alternative relations that may be more

appropriate for this domain. Thus, we will study methods that can automatically identify relations, and potentially increase classification accuracy. Also, our similarity metric is primitive; performance may be improved by assigning and learning the values of attribute weights. Likewise, our collective inferencing algorithm is non-optimal. By eagerly using all the predicted labels in each iteration, if many are wrong, then classification accuracy could suffer. Accuracy may increase if we use a cautious variant of ICA (McDowell *et al.*, 2007b), which would not use low-confidence classification predictions when computing relational attribute values. Finally, collective classification accuracy can be increased by methods that can increase the data's autocorrelation (Aha, 2008), and we plan to test methods with this ability.

This paper describes our initial step towards developing a capability that can assist watchstanders with force protection monitoring tasks. We plan to evaluate our algorithm's utility on additional video of ports, harbors, and other high-traffic maritime areas. In addition, we would like to use additional sensors (e.g., 3-D cameras, infrared, long-range), and, ideally, arrange them on-board to provide 360%, real-time surveillance coverage for use in a variety of conditions (e.g., night, fog, precipitation) in many maritime environments.

## 7 Conclusion

Maritime surveillance for counter terrorism and force protection is manually intensive and error prone due to information overload, fatigue, and imperfect sensors. Although there is a significant opportunity for automated threat analysis from surveillance video, this problem is challenging. For example, image processing techniques may erroneously identify objects, and the low-level sensor data can be noisy.

In this paper, we focused on object recognition, an initial part of the problem of performing automated threat analysis from surveillance video. We took a unique approach to the problem by transforming a maritime scene into a graph of spatially related objects, instead of considering each object independently. This enabled us to represent and exploit the information contained in the contextual cues (i.e., the relations among objects) by applying collective classification algorithms. For one such algorithm, the Iterative Collective Algorithm (ICA), we found that it can significantly increase classification accuracy when using a case-based classifier.

We developed a novel representation for maritime object classification, applied a case-based collective classifier, and empirically demonstrated its utility. We used a domain-specific function for computing the similarity of topological relations.

There are many issues that we plan to address in our future work to improve on the methods presented here. For example, we will explore the use of cautious approaches for collective classification (McDowell *et al.*, 2007b) and other more sophisticated collective inference algorithms (Sen *et al.*, 2008; Aha, 2008). We will also enhance our relational representation to include temporal relations, and assess methods for automatically transforming and selecting relations for our case representation. Finally, we will investigate the use of similarity metric learning techniques.

## Acknowledgements

Thanks to Ralph Hartley for implementing MAAW's image processing software. This research was supported by the Office of Naval Research and NRL.

## References

- Aboutalib, S., Veloso, M.: Towards using multiple cues for robust object recognition. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 189–196. ACM Press, Honolulu (2007)
- Adams, S., Goel, A.K.: A STAB at making sense of VAST data. In: Geib, C., Pynadath, D. (eds.) Plan, Activity, and Intent Recognition: Papers from the AAAI Workshop (Technical Report WS-07-09). AAAI Press, Vancouver (2007)
- Aha, D.W.: Object classification in a relational world: A modest review and initial contributions. In: Proceedings of the Nineteenth Irish Conference on Artificial Intelligence and Cognitive Science, p. 1. Cork, Ireland (Unpublished)
- Burke, R., Kass, A.: Supporting learning through active retrieval of video stories. *Journal of Expert Systems with Applications* 9(5), 361–378 (1995)
- Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Transactions on Pattern Analysis and Machine Intelligence* 6, 721–741 (1984)
- Jensen, D., Neville, J.: Linkage and autocorrelation cause feature selection bias in relational learning. In: Proceedings of the Nineteenth International Conference on Machine Learning, pp. 259–266. Morgan Kaufmann, San Francisco (2002)
- Johnson, C., Birnbaum, L., Bareiss, R., Hinrichs, T.: War stories: Harnessing organizational memories to support task performance. *Intelligence* 11(1), 17–31 (2000)
- Jung, J., Han, I., Suh, B.: Risk analysis for electronic commerce using case-based reasoning. *International Journal of Intelligent Systems in Accounting, Finance, & Management* 8, 61–73 (1999)
- Leu, J.-G.: Computing a shape's moments from its boundary. *Pattern Recognition* 24(10), 949–957 (1991)
- Lipton, A.J., Heartwell, C.H., Haering, N., Madden, D.: Critical asset protection, perimeter monitoring, and threat detection using automated video surveillance (2009) (unpublished manuscript), <http://www.objectvideo.com/products/onboard/whitepapers>
- López de Mantaras, R., McSherry, D., Bridge, D.G., Leake, D.B., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K.D., Keane, M., Aamodt, A., Watson, I.D.: Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review* 20(3), 215–240 (2005)
- MacNeil, R.: Generating multimedia presentations automatically using TYRO, the constraint, case-based designer's apprentice. In: Proceedings of the Workshop on Visual Languages, pp. 74–79. IEEE Press, Kobe (1991)
- McDowell, L.K., Gupta, K.M., Aha, D.W.: Case-based collective classification. In: Proceedings of the Twentieth International FLAIRS Conference. AAAI, Key West (2007a)
- McDowell, L., Gupta, K.M., Aha, D.W.: Cautious inference in collective classification. In: Proceedings of the Twenty-Second Conference on Artificial Intelligence, pp. 596–601. AAAI Press, Vancouver (2007b)
- Micarelli, A., Sansonetti, G.: Case-based anomaly detection. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 269–283. Springer, Heidelberg (2007)
- Murdock, J.W., Aha, D.W., Breslow, L.A.: Assessing elaborated hypotheses: An interpretive case-based reasoning approach. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 332–346. Springer, Heidelberg (2003)



- Neville, J., Jensen, D.: Iterative classification in relational data. In: Getoor, L., Jensen, D. (eds.) *Learning Statistical Models from Relational Data: Papers from the AAAI Workshop* (Technical Report WS-00-06). AAAI Press, Austin (2000)
- Neville, J., Jensen, D.: Leveraging relational autocorrelation with latent group models. In: *Proceedings of the Fifth International Conference on Data Mining*, pp. 322–329. IEEE Press, Houston (2005)
- ObjectVideo. Intelligent video surveillance increases security at seaports, <http://objectvideo.com>
- Pearl, J.: *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufman, San Mateo (1988)
- Perner, P.: Case-based reasoning for image interpretation in non-destructive testing. In: *Proceedings of the First European Workshop on Case-Based Reasoning*, vol. II, pp. 403–409. University of Kaiserslautern, Kaiserslautern (1993)
- Perner, P., Holt, A., Richter, M.: Image processing in case-based reasoning. *Knowledge Engineering Review* 20(3), 311–314 (2005)
- Rhodes, B.J., Bomberger, N.A., Seibert, M., Waxman, A.M.: Maritime situation monitoring and awareness using learning mechanisms. In: *Proceedings of Situation Management: Papers from the Military Communications Conf.* IEEE, Atlantic City (2005)
- Rincón, M., Martínez-Cantos, J.: An annotation tool for video understanding. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) *EUROCAST 2007. LNCS*, vol. 4739, pp. 701–708. Springer, Heidelberg (2007)
- Rosenfeld, A., Hummel, R.A., Zucker, S.W.: Scene labeling by relaxation operations. *Transactions on Systems, Man, and Cybernetics* 6(6), 420–433 (1976)
- Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. *AI Magazine* 29(3), 93–106 (2008)
- Zhang, D., Nunamaker, J.F.: A natural language approach to content-based video indexing and retrieval for interactive e-learning. *Transactions on Multimedia* 6(3), 450–458 (2004)

# Case-Based Reasoning for Situation-Aware Ambient Intelligence: A Hospital Ward Evaluation Study

Anders Kofod-Petersen<sup>1</sup> and Agnar Aamodt<sup>2</sup>

<sup>1</sup> SINTEF ICT,  
S. P. Andersens vei 15 b,  
7465, Trondheim, Norway  
akof@sintef.no

<sup>2</sup> Department of Computer and Information Science,  
Norwegian University of Science and Technology,  
7491 Trondheim, Norway  
agnar@idi.ntnu.no

**Abstract.** Ambient intelligent systems are defined as being able to perceive their environment, being aware of the presence of people and other agents, and respond intelligently to these agents' needs. Today the hardware requirements for achieving these capabilities are met. Earlier work have argued that knowledge intensive case-based reasoning is a feasible method for ambient intelligence. In this paper that argument is supported by testing of an implementation in a hospital ward domain, which shows that despite some issues related to the current implementation the case-based reasoner performs at an acceptable level.

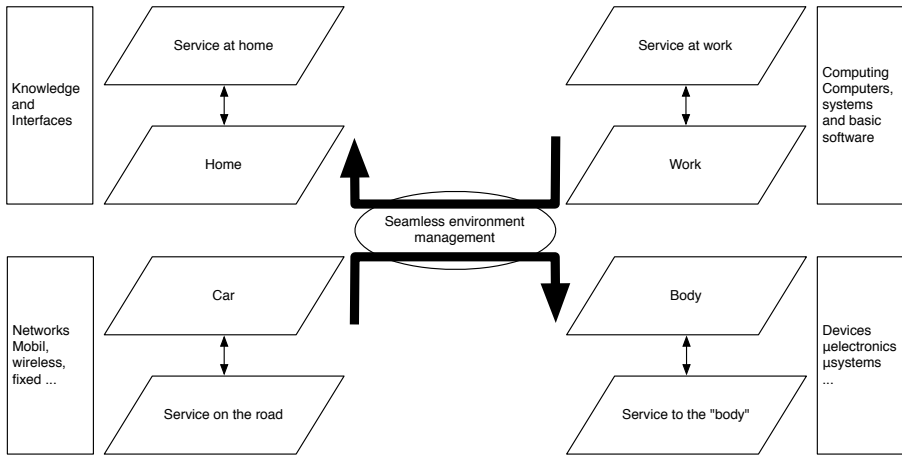
## 1 Introduction

As computers are becoming more *ubiquitous* [1], *pervasive* [2], and *ambient* [4], the need for methods that assist their users in smart and intelligent ways is rapidly increasing. In an ambient system an actual physical computer device will be situated in various physical and social environments at different times, and a challenge is therefore to identify and interpret that environment. Ambient *intelligent* systems, as defined by the ISTAG group [5], are characterised by being able to perceive their environments, be aware of the presence of people and other agents, interpret their own role in that context, and respond intelligently to one or more agents' needs [6].

Realising ambient systems relies on miniaturisation of technology, high calculation power and interconnectivity. According to Satyanarayanan [7], all these hardware tools are currently available. Yet, many of the scenarios described in

---

<sup>1</sup> Recently the term *everywhere computing* [3] has also been introduced. Although all these terms can be viewed as synonyms, a particular term typically indicates a particular perspective, e.g., a physical distributed system perspective vs. a functional-oriented service perspective.



**Fig. 1.** The AmI space (adapted from [9])

the literature still seem like science fiction [8]. When inspecting the literature it appears that the problems of “grasping” everyday life is assumed solved by some sort of intelligent behaviour, but the methods that realise those behaviours are seldom addressed explicitly. The long term vision of the ISTAG group is the concept of *AmI space*, which comprises networked embedded systems that host dynamically configurable services (see Figure 1). The AmI space realises the vision of ambient intelligence by integrating local functionality across a variety of environments. This enables direct, natural and intuitive interaction between the user and services spanning a collection of environments.

In earlier work we have reported on the feasibility of case-based reasoning as a method for realising context-awareness in an ambient intelligent setting [10,11]. In particular, knowledge intensive case-based reasoning, in which the case-based reasoning process is supported by a structural model of general domain knowledge, appears promising. Case-based reasoning lends itself easily to reasoning about context and situations. The fact that case-based reasoning springs from work on understanding reasoning as an explanation process [12], and approaches reasoning by storing and remembering specific episodes (situations), clearly suggests it as a candidate for computational reasoning about situations.

The focus of the work presented here is on the situation awareness task, and more specifically on classifying the situation types in question, based on context elements such as location, time, or type of people involved. Our application domain is the hospital ward domain. The setting is that several times during the day medical personnel gather together for some purpose, with or without the presence of patients. The system’s task is to determine what type of gathering this is. An experimental system has been developed, within the framework of the CREEK system [13,14], and adapted for use on mobile pocket-size computers suitable for being carried around by ward personnel. The system’s architecture,

knowledge structures, and methods are presented in an earlier paper [10], which also exemplifies how a ward situation is classified. In the present paper we briefly review the system architecture and main components, and describe an evaluation study that has been made to assess the system's quality. The system, called AmICREEK<sup>2</sup>, is evaluated according to its ability to classify situations correctly [15]. The results are analysed and discussed within a larger evaluation framework that also include some of the evaluation criteria put forth by Cohen [16].

The rest of the paper is structured as follows: First a short overview of related work is presented. This is followed, in Section 3, by a summary of the architecture of AmICreek. In Section 4 the hospital ward study is presented, with the outcome analysed and discussed in Section 5. A conclusion and outlook on future work ends the paper.

## 2 Related Work

Traditionally, case-based reasoning has been applied to monolithic decision support systems. Recently, however, case-based reasoning has also been applied within the ambient intelligent community. Zimmermann [17] reported on case-based reasoning used to generate recommendations based on the user's context in a mobile environment. The user context was encapsulated inside cases to facilitate comparison of contexts, generating recommendations based on case similarities, and learning of user behaviour. This work was part of a project concerned with audio augmentation of the real world in the context of the art museum in Bonn. Adapting solutions to particular users was also a focus in the work by Ma et al. [18], where case-based reasoning was used to adapt the behaviour of smart homes to users' preferences. Multi-user smart homes can, even with a very limited amount of connected devices, present themselves with a very large amount of possible key processes and dependencies between them. Case-based reasoning was in this work used to identify these interdependencies, due to its ability to reason in ill understood and poorly structured domains. Bénard et al. [19] investigated the use of case-based reasoning as a mechanism for selecting suitable behaviour in different situations. They proposed an agent-based architecture that uses perceived information, or context, as the findings of a case and the proposed action as the solution. The authors ground their context model in a psychological framework that resembles the idea of selective interest and background context as argued by Dewey [20]. This is similar to the approach taken in earlier work within our group [21], which focused on the development of a comprehensive context model. As in that work, Bénard et al. separated context into external and internal context. The former includes the entities present in the environment, and the latter includes skills, states of the

<sup>2</sup> AmICREEK is a recent label for the system, which in earlier reports has been referred to merely as a CREEK system (referring to its framework and CBR method types), or a TrollCREEK system (referring to the Java implementation of CREEK which it was developed from).

agent, the agent's strategies and the agent's history. The existing cases in the case base are pre-classified situations modelled by a domain expert.

Kwon et al. [22] applied case-based reasoning in a multi-agent environment, to estimate the best purchase in comparative shopping. The goal of the system was to achieve a best possible solution between seller and buyer. This system was tested in simulation with several experiments. The three tests were: no negotiation, only price negotiation, and price as well as quality negotiation. As expected by the authors, the complex negotiation outperformed the other two, with respect to buyer's payoff, seller's payoff, and seller's rate of bid winning.

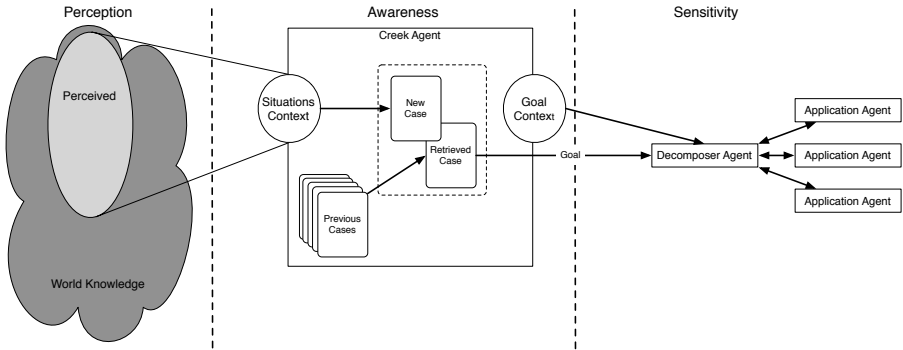
The MyCampus [23] system used case-based reasoning to learn a user's context-sensitive message filtering preferences. When a new message arrived it was compared to the existing messages stored in the case base. The new case was compared to the type of message, the sender of the message, the user's current calendar activity, the user's current location, and the weather. An experiment was carried out to validate the feasibility of the case-based approach. The experiment showed that the accuracy of the filtering process grew from 50 % to over 80 % when using the case-based reasoning approach.

The latter two methods were evaluated by assessing the quality of the best solution, determined by the accuracy of the solution value when compared to the expected value. This is a well-established approach [15], and also the assessment strategy adopted to assess the AmICreek system. As pointed out in [24], however, evaluating the quality of case-based reasoning systems may not be as straightforward, for example when cases are complex structures which are time consuming to construct, or when the user benefits from a larger part of a rich case than just a particular value. These concerns should also make some bells ring when building and evaluating user-interactive systems in health care. Extended approaches to evaluation of case-based reasoning systems include the work of Santamaria et. al [25] addressing the complexity of case-based reasoning applications, the work by McSherry [26] on diversity as a criterion, and the work by Smyth and McKenna [27] on case base competence. Cohen has addressed the evaluation of AI systems in general [28], as well as case-based reasoning systems in particular [16]. In addition to the common accuracy assessment, we relate our result to some of Cohen's criteria in the discussion chapter.

### 3 System Architecture

AmICREEK is an ambient intelligent system that spans a part of the AmI space described in Section 1. AmICREEK observes the ongoing work, thereby becoming *aware of the presence of a person*; assess ongoing situations, thus *perceives the needs* of this person, and *responds intelligently* to ongoing situations. The system is implemented as a three layer architecture where each layer has its own specific responsibility [10]. The AmICREEK architecture is comparable to the model of situation awareness as described by Endsley et al. [29].

The functional architecture is depicted in Figure 2. The main layer of interest in the work presented here is the Awareness layer.



**Fig. 2.** Functional System Architecture

The initial layer is the *Perception* layer, which corresponds to the *Perception* layer in Endsley’s model of situation awareness [29]. This layer is responsible for perceiving the environment and acquiring the necessary knowledge to feed the next two layers. The layer is implemented as a middleware solution that structures gathered information into a coherent structure [30], here named the *Situation Context* (see the middle part of Figure 2). Changes in the Situation Context will trigger an event that will initiate the case-based reasoning cycle.

The second layer is the *Awareness* layer. This corresponds to the second layer in Endsley’s model (*comprehension*), and is the layer that the work presented here focuses on. This layer, which is the layer that exhibits the context-awareness, is implemented using the CREEK method [13,14]. The awareness layer acquires all relevant knowledge from the Perception layer and represents it as the findings of a new case [10]. The new case is matched to the existing case base, the goal is extracted from the best matching case and handed over to the third layer.

The third layer is the *Sensitivity* layer, which is comparable to the third layer in Endsley’s model (*Projection*). This layer acquires the goal identified by the case-based reasoning cycle and constructs a sequence of tasks to execute that will satisfy the goal [31].

Context is used in two different ways within the architecture. Initially context describes the information that is perceivable in the world, labelled as Situation Context in Figure 2. This information constitutes the findings part of the cases, and is important for retrieving similar cases. The second use, labelled Goal Context in Figure 2, is the subset of the Situation Context that is relevant to the goal extracted from a matching case. For a more thorough description of the dualistic use of context and its place in the AmICREEK’s knowledge model, please see [10].

## 4 A Hospital Ward Study

The test presented here tests the Awareness layer, by focusing on the ability of the case-based reasoning system to assess situations correctly. The perception

**Table 1.** Essential Aspects of Situations

Parameter	Description
Location	The room where the situation occurred
User	The user of the system
Role	The role of the user
Present	Other persons present
Role	The role of each of the persons present
Time	The time of day
Source	Information sources and targets
I/O	The direction of the information flow
Information	Type of information
Situation	The type of situation

layer and sensitivity layer have been tested separately on other occasions. The reader is directed to [32] for further details.

#### 4.1 Test Setup

The essential aspects of situations, in the hospital ward domain, are the nine parameters shown in Table 1. These parameters were noted in the ethnographical study detailed in [11]. The topmost six parameters are used to describe the situation as a case and constitute the findings. The next three are used to describe how a goal can be decomposed. Finally, the last parameter is the situation type as classified by the hospital personnel.

For the initial test, the data observed by a human observer, following consultant physician number nine (OL9) were chosen. The physician was the one with the highest number of experienced situations, as well as the one who experienced the highest number of different situations. For the purpose of this test, two days (13 and 14) of observations of OL9 were used. The evaluation of the system's performance was initially done by a qualitative evaluation of the data from the cardiology ward. This was carried out in order to review the context model and the integration of the knowledge model. It is worth noticing that attending one meeting can be viewed as many situations (in sequence). For example, a given meeting of the type pre-ward-round will typically involve the discussion of several patients. Thus, such a meeting is broken down into one pre-ward-round situation per patient discussed.

#### 4.2 Test Execution

The test was conducted as a three-step process. First the 25 different situations that occurred on day 13 were added to the case base, partly as solved and unsolved cases, and the retrieve-step was executed for each of the unsolved cases. Secondly, the learning process of the case-based reasoning algorithm was manually executed. That is, the former unsolved cases were classified and flagged

**Table 2.** Distribution of Observed Situations for OL9, day 13

Situation type	Number of situations	Percentage of all situations covered by day 13	Initial number of solve cases	Sample
Post-work	9	69%	6	1,2,3,4,6,9
Preparation	1	25%	1	1
Pre-ward-round	6	23%	4	2,3,4,6
Patient meeting	1	25%	1	1
Examination	2	25%	1	1
Ward-round	6	23%	4	3,4,5,6

as solved. Finally, all of the day 13 cases were used to classify the situations occurring on day 14.

### 4.3 Initial Modelling

To get a representative distribution of solved and unsolved cases, the 25 different situations for day 13 were divided into two groups.  $\frac{2}{3}$  of the cases (17) were randomly selected and marked as solved, the remaining  $\frac{1}{3}$  (8) were added as unsolved cases. Table 2 describes the number of different situations, the percentages of the total situations, how many were initially selected as solved, and the specific situations in question.

The column labelled *Situation type* describes the six different situations experienced by physician OL9. *Number of situations* counts the number of each situation-type occurring on day 13. The column labelled *Percentage of all situations covered by day 13* shows us how large a percentage the specific situation type that occur on day 13. As an example, the nine observed situations of the type *Post-work* covers 69 % of all observed situations of this type for both day 13 and 14. The column labelled *Initial number of solved cases* gives us the number of cases chosen to be marked as solved. Finally, the column labelled *Sample* tells us which specific situations were modelled as solved cases.

The situations chosen as solved were modelled as cases using the six upper parameters shown in Table 1 as findings. The parameter *Situation* was used to mark each of the cases with a top-level goal, in the sense that for e.g. the *Post-work* situations the goal *Post-work-goal* was identified.

After the modelling of the cases marked as solved the remaining cases were modelled as unsolved. Each of these case were modelled in exactly the same fashion as those solved, except for obviously being marked as unsolved. Each of the unsolved cases were then run through the retrieve-step and matched to the solved cases.

### 4.4 Executing Day 13 Test

Table 3 demonstrates the result of the retrieve process for the unsolved cases of day 13. The current implementation only selects the best matching case for goal



**Table 3.** Result of Matching Test (Run 1)

<b>Input case</b>	<b>Strength Matching case</b>	
Ward round 1301	89%	Pre-ward round 1306
	88%	Pre-ward round 1304
	88%	Pre-ward round 1303
Ward round 1302	88%	Ward round 1305
	88%	Pre-ward round 1306
	88%	Pre-ward round 1304
Pre-ward round 1301	100%	Pre-ward round 1303
	100%	Pre-ward round 1304
	99%	Pre-ward round 1306
Pre-ward round 1305	100%	Pre-ward round 1304
	100%	Pre-ward round 1306
	100%	Pre-ward round 1303
Examination 1302	100%	Examination 1301
	55%	Post work 1304
	54%	Post work 1303
Post work 1305	100%	Post work 1306
	65%	Preparation 1301
	64%	Post work 1304
Post work 1307	99%	Post work 1306
	66%	Preparation 1301
	64%	Post work 1309
Post work 1308	66%	Post work 1309
	65%	Preparation 1301
	61%	Consultation 1304

extraction. So when examining the cases in Table 3 that are classified correctly we can disregard all but the best matching case. Following that line of reasoning only **Ward round 1301** has been misclassified. For **Ward round 1302**, **Ward round 1305** gets selected due to the detailed mechanism for ranking matched cases not apparent in Table 3. AmICREEK suggests that this particular ward round is a pre-ward round, and that the best matching case is **Pre-ward round 1306**.

If we examine the two cases shown, see Figure 3, we will see that they do not look like they should resemble each other to such a high degree as suggested. The differences between the two cases have been marked with dashed-lined circles. Starting with the **Environmental Context**, it contains **Patient #36**, which the matching case does not. Obviously this missing feature should suggest that the two cases differ. The CREEK method does support the ability to lower the matching strength when a matching case misses features. However, the AmICREEK does not yet implement this feature. Looking at the two time values, we can observe that they are very close, and are indeed reported as being 98 % similar. The unknown case occurs in the patient room **PR10**, whereas the best matching case takes place in the **Doctor's office 4**. AmICREEK decides, correctly, that these two locations do not syntactically match. However, both of

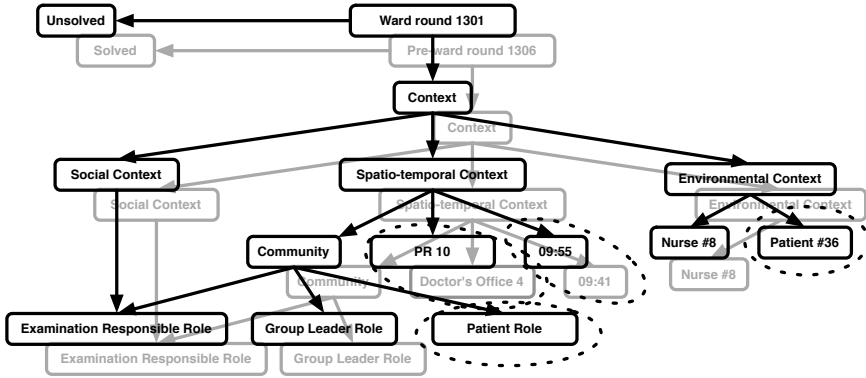


Fig. 3. Ward round 1301 vs. Pre-ward round 1306

these locations are instances of the general class *Location*, and as the knowledge model is structured in a semantic network it should be possible to explain that they are somewhat alike. The CREEK method does allow for this type of reasoning by calculating the convergence point between two concepts [14]. However, currently AmICREEK has been focused on calculation of convergence points between causal relations, thus the calculation for non-causal relations, e.g., *instance of* and *subclass of*, results in convergence between all concepts, as all concepts at some point are an instance of, or subclass of the top-most concept *Thing*. Finally, the unknown case contains a *Patient Role*, not found in the best matching case. This is the same problem as described above regarding the patient in the *Environmental Context*.

If we examine the case base for known cases of the type *Ward round*, the best matching case is *Ward round 1303*, which AmICREEK reports as matching with a strength of 83 %, and as number seven in the list of matching cases. If we examine that case we will discover that it resembles the unknown case a lot. Only two parameters separates them: the time is 09:55 versus 10:13, something which AmICREEK calculates as a 91 % match; and the fact that the patient present is not *Patient #36* but *Patient #38*. It would seem reasonable that these two cases had a high matching strength, however as described above, other cases do not get their matching strength lowered, thus better matching cases cannot compete. This becomes quite evident if we change the patient present to *Patient #36*, in that case the matching strength between the two cases are 99 %, easily surpassing the matching strengths of the cases of the wrong type.

### 4.5 Executing Day 14 Test

As described above, Table 3 shows how AmICREEK classified the cases from day 13 that had been flagged as unsolved. To extend this simulation, we can now move into day 14 by manually executing the learning process, in other words correctly classify these previously unsolved cases and flag them as solved. Table

**Table 4.** Result of Matching Test (Run 2)

<b>Input case</b>	<b>Strength Matching case</b>	
Pre-ward round 1402	100%	Pre-ward round 1303
	100%	Pre-ward round 1301
	100%	Pre-ward round 1304
Pre-ward round 1404	100%	Pre-ward round 1305
	100%	Pre-ward round 1306
	100%	Pre-ward round 1304
Pre-ward round 1407	100%	Pre-ward round 1306
	100%	Pre-ward round 1305
	100%	Pre-ward round 1304
Examination 1404	55%	Examination 1302
	55%	Examination 1301
	54%	Post work 1304
Examination 1405	54%	Examination 1302
	54%	Examination 1301
	54%	Preparation 1301
Ward round 1402	88%	Ward round 1305
	88%	Pre-ward round 1306
	88%	Pre-ward round 1305
Ward round 1405	99%	Ward round 1305
	99%	Ward round 1304
	89%	Pre-ward round 1302
Ward round 1408	89%	Pre-ward round 1302
	88%	Pre-ward round 1301
	87%	Ward round 1305

4 depicts the results of attempting to match the situations from day 14 that was marked as unknown against all the situations from day 13.

If we examine Table 4 we will see that most of the cases were classified correctly. Only Ward round 1408 is classified wrongly, as a Pre-ward round. Looking at Ward round 1408 it matches two pre-ward rounds before Ward round 1305 is matched with a matching strength of 87 %. Again the explanation lies in the fact that semantic matching is currently not implemented and cases are not being “punished” for missing parameters. In this case, the fact that Pre-ward round 1302 is occurring at 11:10 and Ward round 1408 at 11:30 is the one parameter that forces the ranking of the pre-ward round as the best match. As with the example depicted in Figure 3, the pre-ward round neither contains a patient nor a patient role, and the location is wrong.

#### 4.6 Accuracy of the Classifications

The data gathered at the cardiology ward included an *in situ* classification carried out by the physician observed [11]. When measuring the accuracy of the classifications done in AmICREEK we start off from the assumption that these

**Table 5.** Absolute Accuracy of Case-Based Reasoning Test

Situation type	Run 1		Run 2	
	best	all	best	all
Post-work	100%	55%	N/A	N/A
Pre-ward-round	100%	100%	100%	100%
Examination	100%	33%	100%	66%
Ward-round	50%	16%	100%	50%

classifications are correct. Thus, we can compare AmICREEK’s classifications to those carried out by an expert. If we examine Table 5 we can see the absolute accuracy for the four different types of unknown situations that AmICREEK attempted to classify. For both runs the table gives the percentage of times that the best matching case was correct, and the percentage of cases that was correct within the top-three across all matches of this type. The latter can be regarded as the confidence of the classification. Thus, a low number will tell us that the classification was distributed, whereas a high number will tell us that the classification was uniform. As an example, for the **Post-work** cases in *Run 1* all the best matching cases were correct. However, only 55 % of all top-three matches were correct.

As shown in Table 5, AmICREEK does a reasonably good job of classifying the situations correctly. **Post-work**, **Pre-ward-round** and **Examination** are all classified correctly in *Run 1*, and all were classified correctly in *Run 2*<sup>3</sup>. Yet, **Ward-round** situations were only classified correctly 50 % of the time in *Run 1*.

If we look at the distribution among the top-three matches for each situation type we can see that except for situations of the type **pre-ward-round**, most did not classify perfectly. However, as aforementioned, goal extraction was only carried out for the best matching case.

## 5 Analysis and Discussion

The evaluation presented is an evaluation of case-based reasoning as a method for achieving situation-awareness. The use of case-based reasoning as the method of classification is rooted, not only in the desire to find the best suited algorithm for a limited data set, but also in the fact that the theory on human cognition that underlies case-based reasoning is one of situation classification. Thus, the cognitive plausibility [33] of using case-based reasoning is to be found in the general characterisation of case-based reasoning as a way of human reasoning. Hence, the metric for evaluating the use of case-based reasoning in our work, is not its cognitive validity [16], but rather the performances of the classification.

Classification performance can be evaluated in four ways [15]: *i*) absolute accuracy, as determined by a domain expert; *ii*) the plausibility of incorrect

<sup>3</sup> There were no situations of the type **Post-work** occurring on day 14.

classification, also determined by an expert; *iii*) performance can be compared to other algorithms; and *iv*) performance can be compared to that of an expert.

Currently we have no other implementation of an alternative algorithm that fits the data readily available, thus no comparison is available (part *iii* above). Further, as stated above the choice of case-based reasoning as the algorithm of choice is not purely a performance choice, but also one of origin. In addition, the performance of AmICREEK is implicitly already comparable to that of a human expert (part *iv* above). The subjects observed did already do classification during data collection [11], and the classification was *in situ*, thus we must assume that these classifications are correct. For the remaining part of the discussion we focus on the accuracy of the classification and plausibility of incorrect classifications.

Regarding the plausibility of misclassifications, it could be argued that to determine if any misclassifications are plausible an expert working at the particular hospital ward is required. However, given the nature of the observed data, and the amount and distribution of parameters, it seems reasonable that the same assumption regarding the expert with respect to accuracy also holds for misclassifications.

Revisiting the results of the two runs described in Section 4 in light of absolute accuracy, we can re-examine Table 5, we can summarise the accuracy as very good. All possible classifications, except the situation type **Ward-round** were classified correctly. Thus, the absolute accuracy is all in all very good compared to the expert who classified the situations when they were observed.

If we investigate further, we can see that some of the classifications in the top-three matches were not correct. In particular with respect to the situation type **Ward-round**, the situation was misclassified half of the time in *Run 1*. To defend the performance of AmICREEK an explanation for this incorrect classification must be given. Section 4 did already include explanations as to why each of the misclassified cases was classified as it was. The two main general reasons are as follows:

Unsolved cases do sometimes contain features not found in the retrieved case. These extra features should lower the matching strength of the retrieved case. However, the particular implementation of how the matching strength is calculated does currently not take this into account. Thus, the retrieve process uses incorrect matching strengths with respect to this effect. These extra features are very obvious to a human expert, and disregarding them goes a long way in explaining why some cases were misclassified.

Secondly, when a feature is matched in the leaf nodes of the case structure, symbols such as **Location**, **Person** or **Role** are only matched on the syntactical level. This means that the fact that the instances of these concepts are located in the multi-relational semantic network is disregarded in the retrieve process. As with the problem above, the fact that, for an example, a patient is present is a strong indication for a domain expert that a **Ward-round** is taking place. The fact that support for semantic matches of non-causal relations are not yet implemented does explain why some cases were misclassified.

Finally, on day 13 only two **Ward round** cases were initially classified as solved, yet as more cases were solved in day 14 the classification dramatically improved (compare Table 3 and Table 4).

In summary, the situations are being classified at an acceptable level for this first experiment compared to the human domain expert. As we must assume that the classification gathered are correct, the human domain expert does slightly outperform AmICREEK. However, some, if not all of the misclassification can be explained by the way matching strengths are currently being calculated.

## 6 Conclusion and Further Work

The work presented here demonstrates that knowledge intensive case-based reasoning is a promising method for constructing situation-awareness systems. By using the CREEK method, case-based reasoning is applied as the reasoning mechanism to identify ongoing situations. The assessment of situations is the very core of the intelligent behaviour of an ambient intelligent system. It has been demonstrated that using a knowledge intensive case-based reasoning methodology, where the cases are submerged into the general knowledge model, facilitates situation-awareness.

However, it has also been shown that some deficiencies in the current implementation of the CREEK method still exists. In particular with regard to *plausible inheritance*, which currently only works with causal concepts, and with the similarity function that currently does not calculate similarity correctly when cases are missing parameters. Remedying these two issues should significantly improve the system.

Finally, the results should be reproduced using publicly available open source systems. Primarily to verify that it is not the idiosyncrasies of our implementation that allows the reasoning to work, and secondly to allow others to freely use and scrutinise the system.

## Acknowledgements

We would like to extend our gratitude to many of colleagues for their input throughout this work, and in particular to Jörg Cassens, without whom this work would not have been possible.

## References

1. Weiser, M.: The computer for the 21st century. *Scientific American*, 94–104 (September 1991)
2. Hansmann, U., Merk, L., Nicklous, M.S., Stober, T.: *Pervasive Computing: The Mobile World*. Springer Professional Computing (2003)
3. Greenfield, A.: *Everyware: The Dawning Age of Ubiquitous Computing (Voices That Matter)*. New Riders Publishing (2006)

4. Lugmayr, A.: The future is 'ambient'. In: Creutzburg, R., Takala, J.H., Chen, C.W. (eds.) Proceedings of SPIE. Multimedia on Mobile Devices II, vol. 6074. SPIE (2006)
5. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: ISTAG scenarios for ambient intelligence in 2010. Technical report, IST Advisory Group (2001)
6. Aarts, E.H.L., Encarnação, J.L. (eds.): True Visions: The Emergence of Ambient Intelligence. Springer, Heidelberg (2006)
7. Satyanarayanan, M.: A catalyst for mobile and ubiquitous computing. *IEEE Pervasive Computing* 1(1), 2–5 (2002)
8. Lueg, C.: Representation in pervasive computing. In: Proceedings of the Inaugural Asia Pacific Forum on Pervasive Computing (2002)
9. ISTAG: IST advisory group, strategic orientations and priorities for IST in FP6 (June 2002)
10. Kofod-Petersen, A., Aamodt, A.: Contextualised ambient intelligence through case-based reasoning. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) EC-CBR 2006. LNCS, vol. 4106, pp. 211–225. Springer, Heidelberg (2006)
11. Cassens, J., Kofod-Petersen, A.: Using activity theory to model context awareness: a qualitative case study. In: Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference, Florida, USA, pp. 619–624. AAAI Press, Menlo Park (2006)
12. Schank, R.: Dynamic memory; a theory of reminding and learning in computers and people. Cambridge University Press, Cambridge (1982)
13. Aamodt, A.: A knowledge-intensive, integrated approach to problem solving and sustained learning. PhD thesis, University of Trondheim, Norwegian Institute of Technology, Department of Computer Science, University Microfilms PUB 92-08460 (May 1991)
14. Aamodt, A.: Knowledge-intensive case-based reasoning in CREEK. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 1–15. Springer, Heidelberg (2004)
15. Bareiss, R.: The experimental evaluation of a case-based learning apprentice. In: Proceedings of a Workshop on Case-Based Reasoning, Pensacola Beach, Florida, USA, pp. 162–167. Morgan Kaufmann, San Francisco (1989)
16. Cohen, P.R.: Evaluation and case-based reasoning. In: Proceedings of a Workshop on Case-Based Reasoning, Pensacola Beach, Florida, USA, pp. 168–172. Morgan Kaufmann, San Francisco (1989)
17. Zimmermann, A.: Context-awareness in user modelling: Requirements analysis for a case-based reasoning application. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 718–732. Springer, Heidelberg (2003)
18. Ma, T., Kim, Y.D., Ma, Q., Tang, M., Zhou, W.: Context-aware implementation based on CBR for smart home. In: Wireless And Mobile Computing, Networking And Communications (WiMob 2005), pp. 112–115. IEEE Computer Society, Los Alamitos (2005)
19. Bénard, R., Bossard, C., Loor, P.D.: Context's modelling for participative simulation. In: Proceedings of the 19th FLAIRS Conference, pp. 613–618. AAAI Press, Menlo Park (2006)
20. Dewey, J.: Context and Thought. University of California Press (1931)
21. Öztürk, P., Aamodt, A.: A context model for knowledge-intensive case-based reasoning. *International Journal of Human Computer Studies* 48, 331–355 (1998)

22. Kwon, O.B., Sadeh, N.: Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping. *Decision Support Systems* 37(2), 199–213 (2004)
23. Sadeh, N., Gandon, F., Kwon, O.B.: Ambient intelligence: The mycampus experience. Technical Report CMU-ISRI-05-123, Carnegie Mellon University (July 2005)
24. Gu, M., Aamodt, A.: Evaluating CBR systems using different data sources: A case study. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) *ECCBR 2006*. LNCS, vol. 4106, pp. 121–135. Springer, Heidelberg (2006)
25. Santamaria, J.C., Ram, A.: Systematic evaluation of design decisions in CBR systems. In: *Proceedings of the AAAI Case-Based Reasoning Workshop*, Seattle, Washington, USA, pp. 23–29 (1994)
26. McSherry, D.: Diversity-conscious retrieval. In: Craw, S., Preece, A.D. (eds.) *EC-CBR 2002*. LNCS (LNAI), vol. 2416, pp. 219–233. Springer, Heidelberg (2002)
27. Smyth, B., Mckenna, E.: Modelling the competence of case-bases. In: Smyth, B., Cunningham, P. (eds.) *EWCBR 1998*. LNCS, vol. 1488, pp. 208–220. Springer, Heidelberg (1998)
28. Cohen, P., Howe, A.: How evaluation guides ai research. *AI Magazine* (9), 35–43 (1988)
29. Endsley, M.R., Bolté, B., Jones, D.G.: *Designing for Situation Awareness: An Approach to User-Centered Design*. Taylor & Francis, Abington (2003)
30. Kofod-Petersen, A., Mikalsen, M.: Context: Representation and reasoning – representing and reasoning about context in a mobile environment. *Revue d’Intelligence Artificielle* 19(3), 479–498 (2005)
31. Gundersen, O.E., Kofod-Petersen, A.: Multiagent based problem-solving in a mobile environment. In: Coward, E. (ed.) *Norsk Informatikkonferanse 2005, NIK 2005*, Institutt for Informatikk, Universitetet i Bergen, November 2005, pp. 7–18 (2005)
32. Kofod-Petersen, A.: *A Case-Based Approach to Realising Ambient Intelligence among Agents*. PhD thesis, Department of Computer and Information Sciences, Norwegian University of Science and Technology (2007)
33. Cohen, P.R., Howe, A.E.: How evaluation guides AI research. *AI Magazine* 9(4), 35–43 (1988)



# Spatial Event Prediction by Combining Value Function Approximation and Case-Based Reasoning

Hua Li<sup>1</sup>, Héctor Muñoz-Avila<sup>2</sup>, Diane Bramsen<sup>1</sup>, Chad Hogg<sup>2</sup>, and Rafael Alonso<sup>1</sup>

<sup>1</sup> SET Corporation, 1005 N. Glebe Rd.,  
Suite 400, Arlington, VA 22201

{hli, dbramsen, ralonso}@setcorp.com

<sup>2</sup> Department of Computer Science and Engineering, 19 Memorial Drive West,  
Lehigh University, Bethlehem, PA 18015  
{hem4, cmh204}@lehigh.edu

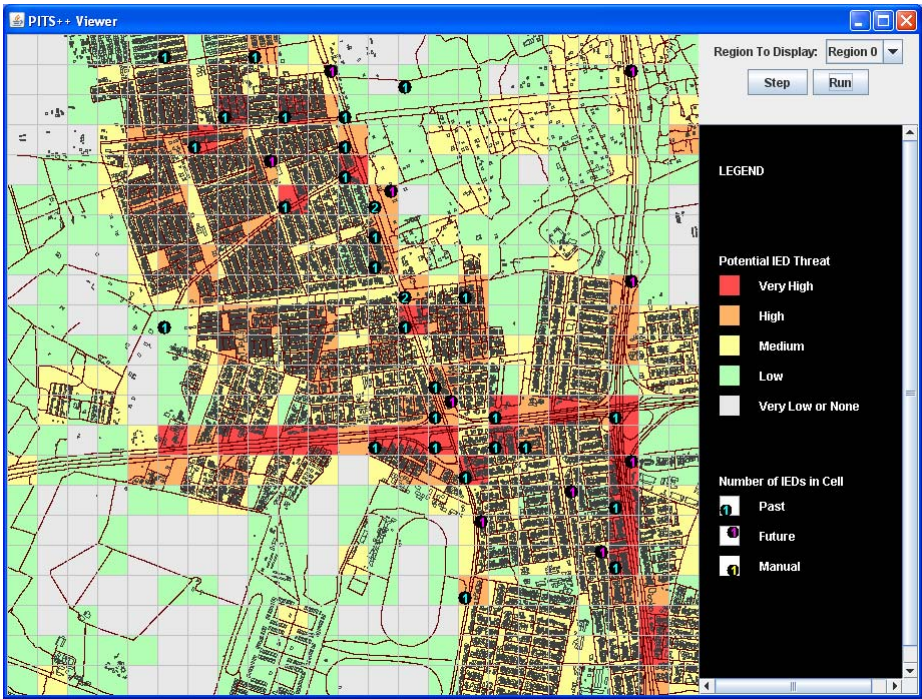
**Abstract.** This paper presents a new approach for spatial event prediction that combines a value function approximation algorithm and case-based reasoning predictors. Each of these predictors makes unique contributions to the overall spatial event prediction. The function value approximation prediction is particularly suitable to reasoning with geographical features such as the (x,y) coordinates of an event. The case-based prediction is particularly well suited to deal with non-geographical features such as the time of the event or income level of the population. We claim that the combination of these two predictors results in a significant improvement of the accuracy in the spatial event prediction compared to pure geographically-based predictions. We support our claim by reporting on an ablation study for the prediction of improvised explosive device (IED) attacks.

**Keywords:** spatial prediction, case-based prediction, function value approximation.

## 1 Introduction

Spatial event prediction is a problem for which the input is a series of events  $e_1, e_2, \dots, e_n$  and their location in a map [1,2,3]. These events have time stamps associated with them, in addition to the locations in the map where they occur and some additional information (e.g., type of event). Based on these locations, regions or influence zones are found. Within an influence zone, cells may have different *influence values*, which are weights associated with cells reflecting a prediction about the potential locations of future events.

Figure 1 presents an example of an influence map generated by the PITS++ system, our function value estimation predictor, for improvised explosive device (IED) attacks in an urban location. PITS++ uses a function value approximation mechanism to update the influence values each time a new IED event is entered into the system. IED attacks are a type of attack where groups of insurgents place an explosive device that is triggered to explode when a target moves close by. These kinds of attacks have



**Fig. 1.** PITS++ viewer showing the training (cyan) and test (magenta) events

become very common in Iraq and elsewhere and are frequently discussed by the news media. The colors are not visible in black and white printout but, basically, we use cyan colored numbers to indicate the locations of IED attacks and the colored-areas indicate the likelihood of attacks. Each cell is colored white (zero likelihood), green (very unlikely), yellow (somewhat likely), orange (likely), or red (very likely).

The PITS++ system is based on its predecessor PITS system [4], which computes these influence maps based on purely geographical features such as the (x,y) location of the map. Our goal is to enhance the influence map by adding non-geographical features such as time or income level in the area of the attacks. To accomplish this we added case-based reasoning capabilities to PITS++ to directly modify or “retouch” the influence values to take into account the contribution of the non-geographical features. The case-based reasoning module, Domain Independent Case-base Assistant) stores copies of the events originally used to train the PITS++ system but also annotated with the non-geographical features associated with the event such as the time of the attack. Then, for each cell in the influence map it retrieves all cases whose similarity to the features of the case is greater than a certain threshold. These cases are then used to retouch the influence values by taking into account the following factors:

- The time stamps from the retrieved cases
- The similarities of the retrieved cases
- The number of cases retrieved

In this paper we will discuss how these factors were combined into a retouching formula and show in an ablation study on synthetic data that, CBR substantially improves the accuracy of the prediction of the PITS++ system. To the best of our knowledge, this is the first time that case-based reasoning approaches have been combined with function value estimation predictors for the task of spatial event prediction. Our results demonstrate the significant impact that CBR can have for this task.

The paper continues as follows: the next section describes the PITS++ system function value estimation predictor; Section 3, the main section of the paper, describes in detail the CBR techniques used to enhance the spatial prediction process; Section 4 discusses the results of our empirical evaluation; Section 5 discusses related work; finally, we make concluding remarks.

## 2 PITS++: Function Value Estimation Prediction

SET Corporation's PITS++ tool dynamically assesses the potential of IED threat, i.e., the likelihood of insurgents emplacing IEDs in a geographic area, by making an estimation of the prediction function value based on a collection of input events.

The PITS++ tool is built on our previous work on PITS [4]. Note that the main difference between the two systems is that PITS++ incorporates a CBR mechanism into the original PITS system. Figure 2 provides an overview of the original PITS system function value estimation mechanism [4]. The inputs to PITS include terrain, IED events, and a history of friendly (blue) and opponent (red) force activity. In PITS the region of interest is a rectangle bounded by geographic coordinates and divided into cells of configurable dimension. PITS extracts IED-relevant features from an input message stream and populates each cell with the terrain and history data relevant to that cell. PITS computes over these IED relevant features to determine the influence value, which we call the PIT value, for each cell. These features (e.g., intersections and corners) are systematically determined using behavioral heuristics as well as knowledge from subject matter experts (SMEs). Each feature has a weight associated with it that indicates the opponent's preference for a feature in the context of IED emplacement activities. The feature weights are dynamically adapted with the latest IED events using function value estimation algorithms [5,6]. The cells are grouped into IED influence regions based on a cell's location and PIT value.

Prediction of IED emplacements is captured by the IED map, which is a grouping of all the IED attractiveness regions in the terrain at a given point in time. Each cell in the grid is thermally colored according to its potential IED threat level. In Figure 1, past, future (during evaluation phases), and manually input (current) IED events are indicated by numbers displayed in the lower left, upper right, and lower right corners of the cells, respectively. As a temporally ordered list of events are entered into the system, the corresponding PIT values are adjusted based on a scalar function on the preferences elicited so far and the features. In Figure 2, The Feature Map lists all the cells that contain non-zero values for each feature. The BattlefieldAOP class is a representation of the area of interest as a grid of cells. It is responsible for populating each cell with the terrain and history information relevant to the cell.

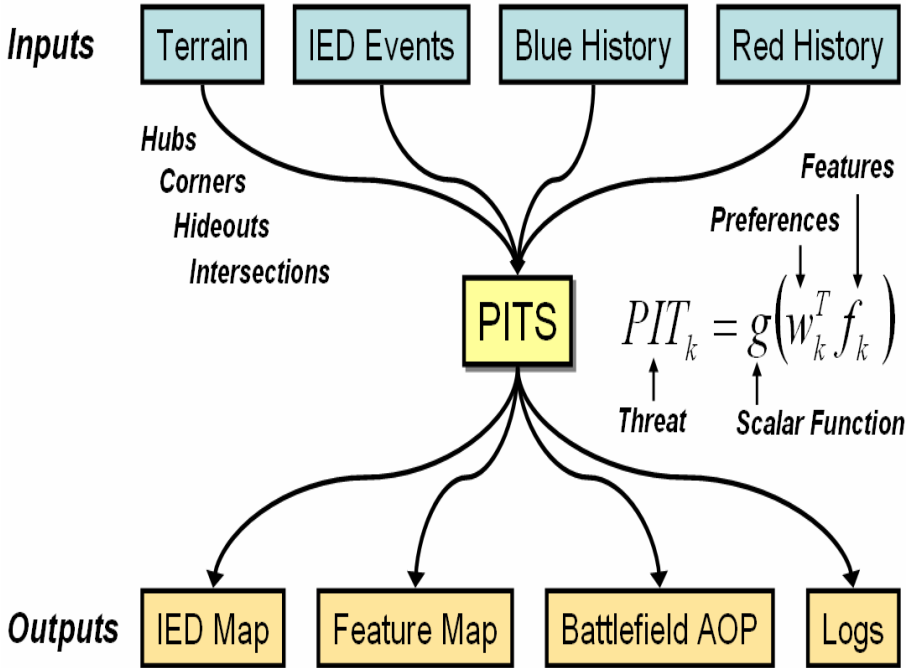


Fig. 2. Function value estimation mechanism of the Potential IED Threat System (PITS)

The following are the geographical features computed in PITS (their values are normalized so they are always between 0 and 1). These features were obtained from interviews with subject matter experts:

- **Roads:** We calculate this value by summing up the number of roads in the cell.
- **Corners/Intersections:** Because of the data, we do not distinguish corners from intersections. To calculate this value, we simply look to see if there is at least one corner or intersection in the cell. If so, the cell gets a value of .5 for this feature<sup>1</sup>. If the cell does not contain a corner or intersection, then the value is 0. Multiple corners/intersections have no additional impact on the feature value.
- **Buildings:** This feature is meant to identify dense areas of the city, so we are looking to see if the cell contains at least 5 buildings. If so, we give it a value of 1 for this feature, and 0 otherwise.
- **Prior IEDs:** If an IED has gone off in the cell, the cell will have a value of 1 for this feature, and 0 otherwise.

<sup>1</sup> For this feature the values are either 0 or 0.5. We did not assign a max value of 1 when corners or intersections were present because this feature was deemed less significant as those with max value of 1, e.g. Roads.

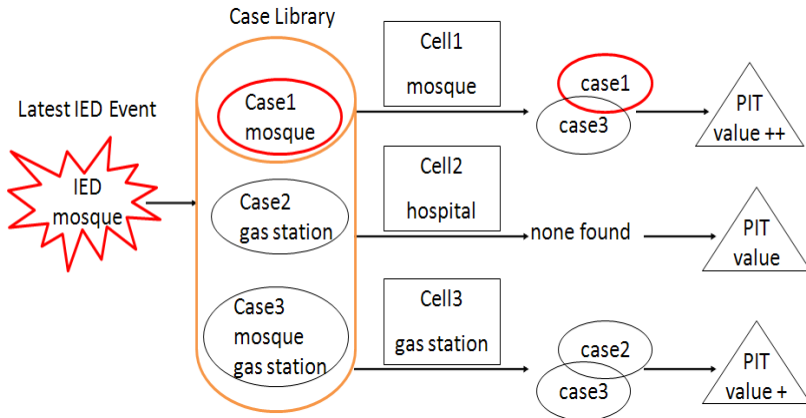


Fig. 3. Retouching PIT values with retrieved cases

### 3 Integrated Prediction with CBR

The basic premise is to update the PIT value (or influence values) by a “retouching” process based on the cases stored in the case base. Retouching works as follows. Suppose the latest IED attack occurred near a mosque. It will be saved in the case library after being processed by the CBR module. At the time of prediction, for each cell in the battlefield grid IED map, a query case will be created using all features associated with this cell. The query case will be dispatched to the CBR module, which will retrieve a list of similar cases. In Figure 3, the case library contains three cases where case1 is more recent than case2 and case3. Cell1 gets two similar cases (case1 and case3) because they all share the fact that they are near a mosque. Cell3 gets two similar cases (case2 and case3) because they are all linked to a gas station. Cell2, on the other hand, failed to retrieve any similar cases because none of the cases in the library is related to a hospital. Both cell1 and cell3 will have their PIT value bumped up because they found similar cases whereas cell2 will not. In addition, cell1 will have a larger increase than cell2 because the former contains a more recent case case1.

For the purposes of using CBR in the context of IED attack prediction, cases represent IED events. Formally, we define a case to be:

$$\text{Case} = (\text{feature}_1, \dots, \text{feature}_n), \quad (1)$$

where  $\text{feature}_i$  includes both geographical features and non-geographical features. Geographical features, such as if the cell contains a major road intersection, are represented in the original PITS system. Non-geographical features are divided into human terrain (e.g., religion) and attack specific features (e.g., the type of explosive used) [7]. Note that non-geographical features are not represented in the original PITS. So a case can be seen as representing a possible correlation between the geographical and non-geographical features.

### 3.1 Retouching Formula

The correlations between geographical and non-geographical features stored in the cases are used to determine how the PIT value is retouched. Specifically, the increment in PIT value is a function of the following factors:

- Date stamp of the cases. Prediction should be influenced by the date when an event took place. An event that occurred one year ago should carry less weight than a week-old event.
- Similarity of the features of the event and the retrieved cases. Prediction should be influenced by the similarity between the cell in consideration and where an event took place. Closer events should carry more weight than those farther away.
- Number of cases retrieved. The more cases are retrieved, the larger the change in the PIT value.

The old PIT value is updated by a factor of the summation of the retrieved cases, factoring in their similarity and their time stamps. We developed the following formula which commits to these three constraints:

$$PITS_{NEW} = PITS_{OLD} + PITS_{MIN,MAX} * \sum_C (SIM(C)/(SIM_{MIN,MAX} * TIME_{NOW,MIN}(C))) \quad (2)$$

Where:

- C is a variable iterating over all retrieved cases
- $PITS_{OLD}$  is the current PIT value for the cell
- $PITS_{NEW}$  is the value we are trying to compute
- $PITS_{MIN,MAX}$  is a scaling factor that determines the relative significance of the original PIT value and the cases. It is currently defined as a factor of a simple linear interpolation of the possible PIT values, ( $PITS_{MAX} - PITS_{MIN}$ ).
- $SIM(C)$  is the similarity between the case and the PITS++ system cell whose value is being retouched
- $SIM_{MIN,MAX}$  is a factor based on the minimum similarity and maximum similarity values of the cases. We currently set it to 1.
- $TIME_{NOW,MIN}(C)$  is a factor based on how close is the case's time stamp,  $TIME(C)$ , to the date when the retouch is done (NOW) and the earliest date (MIN) for which we consider data useful. The closer the time stamp of C to NOW, the smaller the value of  $TIME_{NOW,MIN}(C)$ , which in turn makes the fraction larger. Conversely, the closer it is to MIN the larger the value of  $TIME_{NOW,MIN}(C)$ , which in turn makes the fraction smaller. It is currently defined as a simple linear interpolation:  $(TIME(C) - MIN) / (NOW - MIN)$

### 3.2 Similarity Metric

The similarity metric in DINCAT aggregates local similarities. The local similarities measure how close are two values of the same feature. For example, if a feature represents the (x,y) location in a map, the similarity between two locations can be defined

**Table 1.** Non-geographical features currently implemented

Name	Type	Parameters	Description
TimeIEDAttack	numeric	0, 24	The time of the IED attack
DateIEDAttack	date	MM/DD/YYYY	The date of the IED attack
DeliveryMode	symbolic	boat-borne, animal-borne, collar-bombs, suicide-bombers, platter-chargers, explosively-formed-penetrators, improvised-rocket-assisted-munitions	Classification of the IED by the delivery mechanism as per JCS Pub 1-02
Target	symbolic	US-Armored-Vehicle, Iraqi-Police-Vehicle, US-Contractor-Vehicle, US-Foot-Patrol, Iraqi-Foot-Patrol, Civilian-vehicle, Civilian-foot	Type of target of the IED attack
TriggerMechanism	symbolic	infrared-light-beam, radio-signal, hard-wire, contact	Trigger mechanism used in the IED
Academic	symbolic	academic, non-academic	Indicates if the area of the IED attack is close to an university
Income	symbolic	1, 2, ..., 9	Indicates the income level in the area of the IED attack (1 is lowest; 9 is highest)
Tribe	symbolic	AlDulaim, BaniTamim, Shamar, AlJanabi, Aljubour, Alazza	Tribe in the area of the IED attack. There are more than 100 tribes in Iraq. Current values reflect the fact that in any one area only a few tribes are present.
Religion	symbolic	Shia, Sunni, Christian	Predominant religion in the area of the IED attack. There are more than 10 religions in Iraq. Current values reflect the fact that in any one area only a few religions are present.

as a function of the inverse of the distance between the two locations. Local similarity  $\text{sim}_i()$  for a feature is defined such that it returns a value between 0 (non similar) and 1 (most similar). We define three forms of local similarities depending on the type of feature:

- **Symbolic.** For symbolic features we assign 1 if they are the same and 0 if they are different.

- **Numeric.** For numeric features we assume that the minimum (min) and maximum (max) values are given, and we define the similarity between two values  $X$  and  $Y$  as the inverse of the ratio of the distance between them and the largest possible distance:  $1 - (|X - Y| / (\max - \min))$ .
- **Date.** For date values, we convert them into absolute times measured in hours relative to a fixed date in time. We assign min and max to be the absolute time for the range dates for the events and use the same formula as with the numeric features.

With these local similarities we compare two vectors of features  $\langle X \rangle$  and  $\langle Y \rangle$  by computing the aggregated similarity metric of the local similarities,  $\text{SIM}_{\text{GLOBAL}}()$ , defined as:

$$\text{SIM}_{\text{GLOBAL}}(X_{1..n}, Y_{1..n}) = \alpha_1 \text{sim}_1(X_1, Y_1) + \dots + \alpha_n \text{sim}_n(X_n, Y_n), \quad (3)$$

where the values of the vector weights,  $\alpha_1 + \dots + \alpha_n$ , sum to 1. As a result,  $\text{SIM}_{\text{GLOBAL}}()$  also returns a value between 0 and 1 (1 been most similar). For our current implementation we set each  $\alpha_i$  to  $1/n$ .

### 3.3 Non-geographical Features

A non-geographical feature may take a date, a numeric value (between a minimum value and a maximum value) or a symbolic value (from a predefined set). For each feature (see Table 1) we identify the following elements:

- **Name:** indicates the name of the feature
- **Type:** indicates the type of the feature; this can be symbolic, numeric, or date.
- **Parameters:** for numeric features this will indicate the minimum and maximum value and for symbolic features this will indicate the set of possible values.
- **Description:** a description of the feature

## 4 Empirical Evaluation

The purpose of the experiment is to evaluate the contributions, if any, of the CBR approach to the event spatial prediction made by PITS. For this purpose we performed an ablation study where we compared the results of PITS versus PITS++ (PITS augmented with the CBR retouch mechanism) on the same data. As mentioned before, cases can be seen as storing information of previous events co-relating geographical and non-geographical features. Therefore, it is conceivable that using our CBR retouching approach will result in improvements in the prediction when, as a whole, correlations exists between the geographical and non-geographical features. However, it might be detrimental to use the CBR approach when no such co-relations exist. Therefore, we created 3 data sets to observe the performance in 3 scenarios:

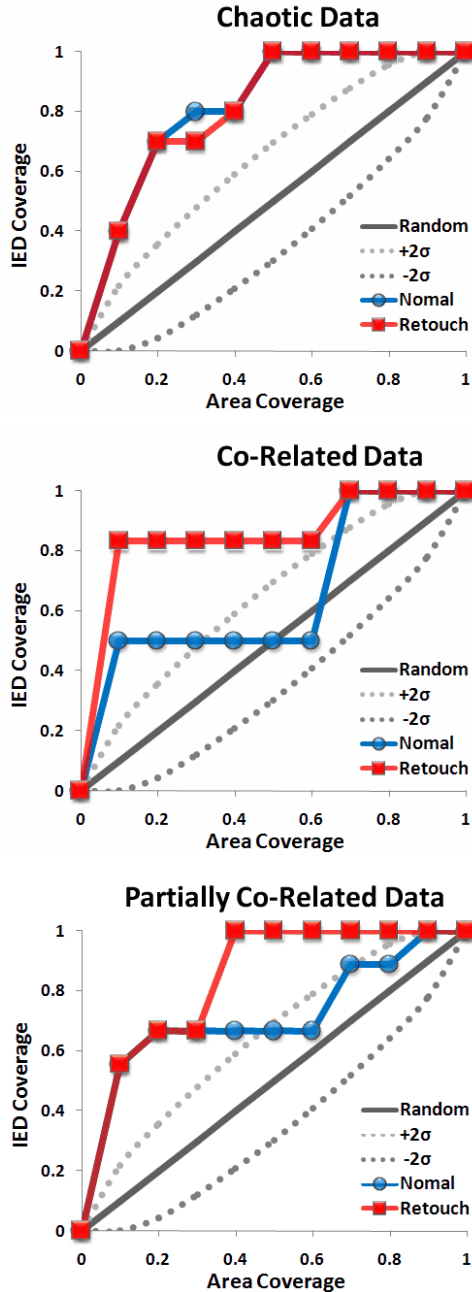
- **Correlated.** The data set consisted of 48 events in which 3 kinds of correlations exist between the geographical and non-geographical features. Every event in the data set commits to one of these correlations.



- Partially correlated.** The data set consisted of 44 events in which 5 kinds of correlations exist between the geographical and non-geographical features. Roughly 1/4 of the events have no correlation whatsoever.
- Chaotic.** The data set consists of 42 events. No co-relations exist between the geographical and non-geographical features.

Each data set was divided into a training set consisting of 3/4 of the data and a testing set consisting of the remaining 1/4 of the data. The retrieval threshold was set to 0.75.

The pseudo-ROC (Receiver Operating Characteristics) curve was used to evaluate the PITS++ system's performance as an IED event predictor (Figure 4). It is defined as the IED coverage, which is the percentage of future IED events covered by threat regions, plotted as a function of the area coverage, which is the percentage of the playbox (i.e., area of prediction) occupied by threat regions. The threat regions are determined by the PITS++ IED threat values. An area coverage of 10% looks at the top 10% of cells that have the



**Fig. 4.** ROC curves for the chaotic (top), the correlated (middle), and the partially correlated (bottom) data sets with similarity threshold at 0.75

highest IED attractiveness value. In general, the more rapidly the pseudo-ROC curve rises, the better the predictor.

A random predictor has a curve close to the diagonal, which is drawn in the plots as a solid black line. We also plot the curves that mark 2-standard deviations from the random predictor as dotted lines around the diagonal (Figure 4). For a predictor to be statistically better than the random predictor, its curve should be above the upper dotted line.

Figure 4 shows the resulting ROC curves for the chaotic, partially correlated, and correlated data sets. For each of these, two figures are drawn: one indicating the results with retouch and for PITS without any retouch. In addition we present the curve  $Y = X$ , which indicates a random prediction. The X-axis denotes the area coverage for the prediction. So 0.1 represents a prediction within 10% of the map whereas 1 represents all area. The Y-axis denotes the accuracy of the prediction. So, for example, the random predictor achieves 100% prediction only when 100% of the area is covered. Therefore, the results when comparing the two curves are particularly interesting for lower values of X or at the very least to the first X-point where  $Y=1$  is achieved. Overall a good comparison of the performance is the area under their curves. We compare the ratio:  $\text{area}(\text{CBR})/\text{area}(\text{non-CBR})$ .

The results are as follows: for the correlated dataset the curve for the CBR is always above the non-CBR approach until  $X = 0.7$ . The ratio of the CBR over the non-CBR is 77.7% if we consider only the areas until both curves reaches  $Y = 1$  (i.e.,  $X = 0.7$ ). The ratio reduces to 66.6% if we count all area between  $X = 0$  and  $X = 1$ . For the partially correlated, the performance is the same for both CBR and non-CBR until  $X = 0.3$ . Between 0.3 and 0.9 the CBR performance improves over the non-CBR and they are tied again between 0.9 and 1. The ratio of the CBR over the non-CBR is 86.2% and it augments to 82% if we consider only the areas until both curves reaches  $Y = 1$  (i.e.,  $X = 0.9$ ). Although it is noteworthy that with CBR it reaches 1 at  $X = 0.4$  whereas non-CBR reaches 1 only at  $X = 0.9$ . Finally for chaotic both curves are the same and only in the interval  $[0.2, 0.4]$  the non-CBR does slightly better. However, the ratio of the non-CBR over the CBR is only 98.8% and it augments to only 97.2% if we consider only the areas until both curves reaches  $Y = 1$  (i.e.,  $X = 0.5$ ). We made only one run with each data set because the system behaves deterministically: with the same input sequence of events, PITS and the CBR retouching algorithm produces the same values. The random predictor may produce multiple values, but on average it produces the  $y=x$  curve shown in our analysis.

We ran student t-tests on the results obtained from the experiments comparing the data points for the normal versus the retouched results. The difference in score for the chaotic data set is not significant (t-test score: 87%), for the correlated data set is significant (t-test score: 99.7%), and for the partially correlated data set is significant (t-test score: 98.6%). In conclusion, if there is a co-relation between the geographical and the non-geographical features, -even if only partial- the CBR retouch will improve the performance of the prediction. If there is no co-relation the CBR retouch would have a negligible negative impact on the prediction.

## 5 Related Work

Related approaches can be divided into three kinds: time prediction from time series, prediction from influence maps, and case-based prediction. We briefly discuss each insofar as to contrast with our approach. Each of these has been the subject of extensive and well-established research. The problem of prediction from time series in its most simple form can be defined as to obtain a time range prediction  $[t, t']$  for an event from a history of event in time  $t_1, t_2, \dots, t_n$  [8]. This is a well-founded field. Some methods assume an implicit model for time series while others assume an explicit model. The problem of IED time prediction is dependent on geographical, human terrain, and attack-specific features, for which, to the best of our knowledge, no time prediction model exists capable of incorporating all of these kinds of features.

Influence maps is a method for spatial analysis which receives as input a series of events  $e_1, e_2, \dots, e_n$  and their location in a map [1,2]. These events do not necessarily have time stamps; just the locations in the map where they occur and some additional information (e.g., type of event). Based on these locations, regions or influence zones are found. Within an influence zone, cells may have different weights reflecting more or less influence from the events in the cell. These weights are typically represented with colors for visualization purposes. In influence maps the geographical location (e.g., the "x,y" coordinates) play a significant role in how the regions are determined. Whereas indeed the graphical locations are important, our work aims to find common geographical features between attacks rather than just the "x,y" location. Moreover, we want to incorporate non-geographical features (i.e., human terrain and attack-specific) into the process of determining these regions. Again, to the best of our knowledge no work exists accomplishing this.

Case-based prediction refers to the use of case-based reasoning (CBR) as the prediction technique [9,10,11,12,13,14]. Predictions in CBR include time prediction as well as class prediction (e.g., predict the kind of object based on partial observations), and strategy prediction (e.g., predicting the next movement from an opponent) among others. The difficulty of using CBR for IED prediction is that events might be related at different and contrasting levels (e.g., they might be geographically close but rather different from the perspective of the human terrain). Therefore, instead of tackling the whole problem with CBR we aim at using CBR for making predictions with the non-geographical features and combine this prediction with the one from the PITS system which models the geographical features. Analogously, adding all features to the PITS model, which is in essence the approach taken in Liu and Brown [3], would introduce the curse of dimensionality. This is a well documented limitation of value function approximation algorithms [15].

Case-based reasoning has been combined with value function approximation algorithms such as reinforcement learning [16,17,18,19] and neural networks [20,21]. The particular value function approximation algorithm developed in PITS++ has been shown to be particularly useful for spatial event prediction and, therefore, a suitable base line to measure performance gains by using case-based reasoning techniques.

## 6 Conclusions

One major aspect of defeating the use of IEDs is defeating the device itself. There has been intensive effort devoted to combining sensor data of various types to identify those who planted the IEDs. There are several limitations in those approaches. First, it typically tries to identify the emplacements after the attack. In other words, the current work is post-mortem and reactive in nature rather than predictive and proactive. Second, this work has been heavily relying on a human expert to perform the historical pattern analysis and recognition. There is an outcry for predictive algorithms that operate in an autonomous or semi-autonomous manner. Third, this work has limited ability to fuse all potential data sources (geospatial and temporal event information, social and cultural data, coalition traffic patterns, multi-spectral sources of sensor data, etc). Existing methods could be used by adding all necessary features into a given value function approximation algorithm. However, this will incur the curse of dimensionality.

We presented an alternative approach in PITS++ for spatial event prediction that combines function value estimation and a case-based prediction. PITS, the function value estimation predictor, contributes with its capability to reason with geographical features such as the  $(x,y)$  coordinates of an event. We enhance this capability by using case-based reasoning techniques to model non-geographical features such as the time of the event or trigger mechanism of the IED. Cases capture, in essence, instances of correlations between geographical and non-geographical features. This correlation is exploited by our system to update the geographical prediction of PITS. We observed that this update results in a significant improvement of the accuracy in the empirical prediction compared to pure geographically-based predictions when there is some correlation between the geographical and the non-geographical features in the input event traces. In the worst scenario, when no such a co-relation exists, CBR does not help but it is not detrimental either. We support these claims with an experiment on an ablation study for the prediction of improvised explosive device (IED) attacks.

There are several directions that we want to explore in the future. First, there was little tuning in the CBR component. In particular, each local similarity metric was assigned the same weight. It is conceivable that further improvements in the prediction performance can be achieved if weights are tuned by, for example, performing statistical analysis of the data. Second, the integration between the value function approximation and the CBR prediction is one way; the CBR retouching process does not permanently change the PIT values. Instead, it is currently designed to perform the CBR retouch on the PIT values every time a new prediction is needed. We need to investigate what would be the implication of keeping the retouched PIT values; particularly what does this mean for the value function approximation process. Third, we would like to perform an evaluation with real data. For the current evaluation we used simulated data because of the unavailability of adequate real data at the time of the study. Although public data exists about IED attacks, this contains mostly geographical features. Fourth, all features currently used in the study were manually given based on interviews with Subject Matter Experts. A potential research direction is to learn such features by extracting them from raw IED data (e.g., from a repository of field reports). Fifth, the values of parameters in the retouch formula (Formula 2) are currently set manually. A potential research direction is to use Bayesian or other

methods to learn these parameters. Sixth, we would like to investigate time prediction of IED events, building on existing work for explicit representations and reasoning methods for temporal events.

## Acknowledgements

This research was supported by grants from the Air Force Research Laboratory and the National Science Foundation Grant No. NSF 0642882. The opinions stated in this paper are those of the authors and not of the funding agencies supporting this work.

## References

1. Sweetser, P.: Strategic Decision-Making with Neural Networks and Influence Maps. AI Programming Wisdom 2, Charles River Media (2004)
2. Tozour, P.: Influence Mapping. In: DeLoura, M. (ed.) Game Programming Gems 2, pp. 287–297. Charles River Media (2001)
3. Liu, H., Brown, D.E.: Spatial-Temporal Event Prediction: A New Model. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, San Diego, California (October 1998)
4. Li, H., Bramsen, D., Alonso, R.: Potential IED Threat System (PITS). In: Proceedings of 2009 IEEE International Conference on Technologies for Homeland Security (HST 2009). IEEE Xplore, Los Alamitos (to appear, 2009)
5. Alonso, R., Bloom, J.A., Li, H., Basu, C.: An Adaptive Nearest Neighbor Search for a Parts Acquisition ePortal. In: Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining, SIGKDD 2003 (2003)
6. Alonso, R., Li, H.: Model-Guided Information Discovery for Intelligence Analysis. In: Proceedings of CIKM 2005, Bremen, Germany (2005)
7. Kipp, J., Grau, L., Prinslow, K., Smith, D.: The Human Terrain System: A CORDS for the 21st Century. Military Review (September-October 2006)
8. Hamilton, J.D.: Time Series Analysis. Princeton University Press, Princeton (1994)
9. Faltings, B.: Probabilistic Indexing for Case-Based Prediction. In: Leake, D.B., Plaza, E. (eds.) ICCBR 1997. LNCS, vol. 1266, pp. 611–622. Springer, Heidelberg (1997)
10. Hansen, B.K.: Weather prediction using case-based reasoning and fuzzy set theory. M.S. thesis, Dept. of Computer Science, Technical University of Nova Scotia (2002)
11. Jære, M.D., Aamodt, A., Skalle, P.: Representing Temporal Knowledge for Case-Based Prediction. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416, p. 174. Springer, Heidelberg (2002)
12. Zehraoui, F., Kanawati, R., Salotti, S.: Case base maintenance for improving prediction quality. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689. Springer, Heidelberg (2003)
13. Redmond, M., Line, C.B.: Empirical Analysis of Case-Based Reasoning and Other Prediction Methods in a Social Science Domain: Repeat Criminal Victimization. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689. Springer, Heidelberg (2003)
14. Weber, R., Evanco, W., Waller, M., Verner, J.: Identifying critical factors in case-based prediction. In: Proceedings of the 18th FLAIRS Conference. AAAI Press, Menlo Park (2004)
15. Bellman, R.E.: Adaptive Control Processes. Princeton University Press, Princeton (1961)

16. Bridge, D.: The virtue of reward: Performance, reinforcement and discovery in case-based reasoning. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS, vol. 3620, p. 1. Springer, Heidelberg (2005)
17. Gabel, T., Riedmiller, M.: Multi-agent Case-Based Reasoning for Cooperative Reinforcement Learners. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416. Springer, Heidelberg (2002)
18. Sharma, M., Holmes, M., Santamara, J.C., Irani, A., Isbell Jr., C.L., Ram, A.: Transfer learning in real-time strategy games using hybrid CBR/RL. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). AAAI Press, Menlo Park (2007)
19. Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing The Enemy: Combining Reinforcement Learning with Strategy Selection using Case-Based Reasoning. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 59–73. Springer, Heidelberg (2008)
20. Chen, D., Burrell, P.: Case-based reasoning system and artificial neural networks: A review. *Neural Computing and Applications* (2001)
21. Fdez-Riverola, F., Corchado, J.M., Torres, J.M.: An Automated Hybrid CBR System for Forecasting. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416, p. 522. Springer, Heidelberg (2002)

# Case-Based Support for Forestry Decisions: How to See the Wood from the Trees

Conor Nugent<sup>1,\*</sup>, Derek Bridge<sup>2</sup>, Glen Murphy<sup>3</sup>,  
and Bernt-Håvard Øyen<sup>4</sup>

<sup>1</sup> Idir Technologies, Ireland  
nugentc@gmail.com

<sup>2</sup> Department of Computer Science,  
University College Cork, Ireland  
d.bridge@cs.ucc.ie

<sup>3</sup> Forest Engineering, Resources and Management Department,  
Oregon State University, Corvallis, Oregon, USA  
glen.murphy@oregonstate.edu

<sup>4</sup> Norwegian Forest and Landscape Institute,  
Bergen, Norway  
bernt-havard.oyen@skogoglandskap.no

**Abstract.** In forestry, it is important to be able to accurately determine the volume of timber in a harvesting site and the products that could potentially be produced from that timber. We describe new terrestrial scanning technology that can produce a greater volume of higher quality data about individual trees. We show, however, that scanner data still often produces an incomplete profile of the individual trees. We describe Cabar, a case-based reasoning system that can interpolate missing sections in the scanner data and extrapolate to the upper reaches of the tree. Central to Cabar's operation is a new asymmetric distance function, which we define in the paper. We report some preliminary experimental results that compare Cabar with a traditional approach used in Ireland. The results indicate that Cabar has the potential to better predict the market value of the products.

## 1 Introduction

Forest planners are responsible for deciding how the set of commercially-cultivated forests that are under their control should be developed and eventually harvested. At any given time, a number of different forests are available, and planning how best to utilize them can be a difficult task. For example, forest planners must combine information that comes from processing plants (e.g. sawmills) with information about their forests to decide which forests to use and which trees within those forests to fell.

---

\* This work was carried out while the first author was a member of the Cork Constraint Computation Centre (4C) at University College Cork. The project was an Innovation Partnership (IP/2006/370/), jointly funded by Enterprise Ireland and TreeMetrics, a company that provides forestry measurement systems.

Poor decisions in this planning process are commonplace. But they are not necessarily due to poor judgement. They are often caused by inadequate information at the root of the supply chain, the forest. Forests exhibit high inherent spatial variability (e.g. two trees growing side by side may exhibit different characteristics due to differences in genetics and micro-climate) and temporal variability (e.g. trees continue to grow after being measured). In the vast majority of cases the characteristics of a forest are only vaguely known. This means that resources that have been cultivated over periods of thirty to more than a hundred years are often underutilized based on ill-informed decisions made quickly at the end of their life cycles [1,2,3,4]. These poor decisions and the lack of quality forest information naturally have knock-on effects right through the supply chain.

In this paper, we focus on three forest planning tasks:

**Tree taper estimation:** The task here is to estimate the diameter of a tree stem at different heights. Diameters tend to taper, i.e. they decrease with height, and this is why this is known as taper estimation.

**Stem volume estimation:** The task here is to estimate the volume of timber that a tree will produce. This may be based on estimates of tree taper.

**Product breakout estimation:** The task here is to estimate what products can be produced from a tree, e.g. size and quality of planks, amount of wood-chip, and so on. This may be based on estimates of stem volume.

The rest of this paper is structured as follows. Section 2 presents state-of-the-art methods for the tasks listed above; it explains the role of new scanner technologies; it motivates the use of Case-Based Reasoning (CBR) to exploit the scanner data; and it describes related work in CBR. Section 3 presents Cabar, our case-based reasoning system for the tasks listed above. The focus in the section is on case and query representation, along with a new asymmetric distance function that we have defined. Section 4 describes experiments we have conducted and presents our preliminary results.

## 2 The State of the Art: Motivating the Use of CBR

### 2.1 Current Practices

Much of current practice revolves around the prediction of the expected volume of a forest, i.e. the amount of timber a forest is expected to yield. This aids the selection of which of a set of forests to harvest.

In Ireland, a common approach is to take a set of measurements about a forest and use them to access a simple set of look-up tables that translate these measurements into volume figures. The forest manager conducts a survey of the forest, which records the diameter at breast height (DBH) and, perhaps, the height of a number of sample trees [1]. From these measurements, and perhaps also

<sup>1</sup> In some parts of the world, the survey may also include assessments of stem shape, curvature, and quality (e.g. size of branches, scarring, rot, wood density, etc.) [5].



the forest age and the thinning strategy, the manager can then read-off predictions of the expected volume and, sometimes, the typical dimensions of saw-logs that the forest can yield. The look-up tables are compiled from extensive field measurements and mathematical models. The disadvantages of this approach include: the tables that are available to the manager may not adequately reflect local conditions (soil, weather, tree species, species mixture, etc.); the predictions are made from only a sample of trees in the forest and from only one or two measurements about each tree; and the prediction is only a crude estimate of overall forest volume, and not individual tree volume.

An alternative is to predict volumes on a tree by tree basis. This is usually done by predicting the diameter of the tree stem at different heights along the stem, from which the volume of the tree can be calculated. The equations for predicting diameters are known as taper equations. For the most part, the input parameters are the DBH and the height of the tree [6,7]. In many cases, the height is not measured; rather, it is estimated from the DBH using a height model [8,9]. Many different taper equations exist, each making different assumptions about tree shape, and hence using different geometrical principles and mathematical functions. It is necessary to choose the right equation for the species of tree and to calibrate the equation based on local conditions and historical data. The disadvantages of this approach include: the equations that are available to the manager may not adequately reflect local conditions; and the equations use only small amounts of data about the tree (sometimes just the DBH).

From full taper and volume predictions, it is possible to estimate the product breakout, i.e. the products that might be produced from a tree [10]. Some forest managers use software to do this. The software simulates the algorithms that are used in the field by harvesting machines, as explained below.

Trees are rarely transported from the forest as complete units. They are usually first cross-cut into smaller units (logs). The harvesting machine's on-board software decides how to cross-cut a tree. Obviously, its decisions have a major bearing on what products the sawmill will ultimately be able to produce. The harvester is pre-loaded with data about the products to be cut and their priorities (in the form of a set of weights). The harvester begins by taking hold of the base of a tree; it then both measures and infers the dimensions of the tree; and it uses a priority cutting list or a mathematical programming technique (e.g. dynamic programming [11], branch-and-bound [12], network analysis [13]) to determine the optimal or near-optimal way to cross-cut the tree.

By simulating the harvesting machine's decisions in advance on taper and volume predictions, a forest manager can decide which trees to harvest. But the disadvantages include that poor predictions of taper and volume may render estimates of product breakout too unreliable to be useful in practice.

## 2.2 New Technologies

New technologies offer the potential to overcome the lack of information about forests. They may enable us to obtain a greater volume of higher quality data, and to do so at low cost.

In our research, for example, we have been working with a company called TreeMetrics, who provide forestry measurement systems ([www.treemetrics.com](http://www.treemetrics.com)). TreeMetrics has developed a portable 3-dimensional terrestrial laser scanning technology [14,15]. Their technology makes it possible to capture 3D data about standing trees in a forest prior to harvesting. For each individual tree in a scanned plot the scanner can record hundreds of laser readings. TreeMetrics' software takes a set of readings, tries to work out which tree each reading belongs to, and calculates tree diameters at fixed intervals along the length of the tree. For each diameter, the centre point is also calculated and so curvature information about the tree is also obtained. Such information makes it possible to accurately determine the volume and a quality attribute (from the curvature data) of trees in a forest in advance of harvesting.

### 2.3 A Role for Case-Based Reasoning

Although TreeMetrics' technology provides vastly more tree information than some more traditional approaches, it is not guaranteed to provide a complete picture of each tree. Readings for some sections of a tree may be missing due to occlusion by branches or other trees. This becomes increasingly common the further up the tree the readings are sought due to the effects of branching and the limitations of the laser at increased distances.

This leaves us with a tree profile prediction task, both in terms of interpolating the missing sections and also in terms of extrapolating to the upper reaches of a tree. There is also the task of smoothing or replacing diameter readings which contain noise. In this paper we outline a Case-Based Reasoning (CBR) approach which accomplishes these tasks.

We expect various advantages to accrue from the combined use of better data in greater volumes and the use of CBR in place of pre-compiled look-up tables and equations. These advantages include:

**Flexibility:** As described in Section 2, in Ireland the traditional approach is to make predictions from measurements taken at fixed points, such as the diameter at breast height. Even when more measurements are available, such approaches cannot capitalize on the extra data. Equally, they cannot be used to make predictions if the data they need is not available. The CBR system that we propose can make predictions based on whatever readings are available. In particular, the new similarity measure that we define (Section 3.2) handles any number of readings, and accommodates information about the certainty of those readings.

**Localized predictions:** In traditional approaches, models (e.g. systems of equations) can be calibrated to local circumstances. However, this is complex and the costs of doing it are typically high. Hence it is common to generalize over broad regions, which means that the models often fail to capture finer-grained local variations. CBR offers an approach that can be readily localized. The case base used for Sitka Spruce forests in southwest Ireland, for example, need not be the same as the case base used for Norway Spruce

forests in southeastern Norway. Case bases can be built from harvester data or field measurements that come from the particular area in which the case base will be used, thus implicitly capturing the local characteristics of that area. This is not cost-free, but it is simpler than model calibration.

**Immediate use of data:** With CBR, there is no need for model calibration. In some sense, calibration is implicit: the particular cases in the case base calibrate the system to its local circumstances. Equally, since CBR is a strategy for both problem-solving and learning, by judicious case base update, case bases can be tailored to local conditions over time.

## 2.4 Related Work

Applications of CBR in forestry, while few in number, have a long history. In 1997, Goodenough et al. [16] described the SEIDAM system that tries to keep forest inventories up-to-date through the integration of images and digital maps. Their use of CBR is quite different from ours: they use it to form plans of map update operations. In 1998, Kelly and Cunningham [17] investigated an algorithm for selecting an initial case base from a database of Irish forestry data. Our data is about individual trees, whereas the records in their database provide information about forest ‘sub-compartments’. Their CBR system is judged by how well it predicts the proportion of a sub-compartment that should be planted with a particular species. There are also a number of examples of problems within forestry, including the problem of estimating taper on unmeasured portions of a felled tree stem while the stem is being harvested, for which solutions based on  $k$ -NN have been suggested [18,19,20,4].

## 3 Cabar: A CBR Forestry System

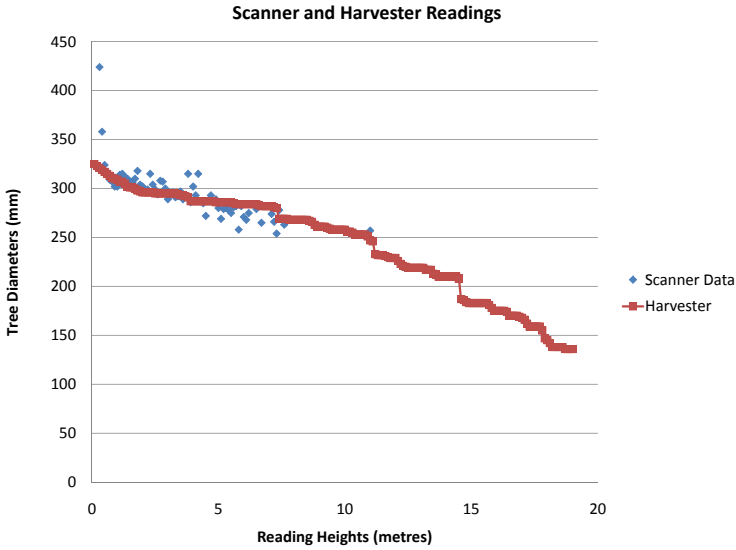
Cabar is the name of the CBR system that we have developed.<sup>2</sup> Cabar is designed to deal with the particular characteristics of tree stem data, especially the kind of data that we can obtain from TreeMetrics’ laser scanner technology. The emphasis is on taper prediction, which we then use for stem volume prediction, which we use in its turn for product breakout estimation. Cabar’s design and operation are explained in the next three subsections.

### 3.1 Cases and Queries

Each case in our case base represents one tree and contains a sequence of real values, which denote the diameter of the tree, usually at 10 cm intervals along its entire length. Sometimes cases come from manual field measurements. But they are also readily available from harvester machines. The on-board software records an overbark profile of each tree that the harvester cuts. There is no solution part to the cases.

---

<sup>2</sup> “Cabar” is the Gaelic word, often spelled “caber” in English, for a wooden pole.



**Fig. 1.** Scanner readings and harvester measurements of the diameter of a tree

Queries in our system are also described by sequences of real values, denoting the diameter of the tree at different heights. Whereas cases may be standing trees that we have measured manually or trees that a harvester has previously felled and measured, queries usually describe standing trees whose taper and volume we wish to predict. They have been scanned, and we have a set of readings from the laser scanner. Hence, we say that cases contain *measurements* and queries contain *readings*.

However, as mentioned above, due to occlusion and the limitations of the scanning technology, a query is often a fairly incomplete impression of the tree profile. This is illustrated in Figure 1, which shows the profile of a tree as recorded by TreeMetrics' scanning device (diamonds) and the profile of the same tree as recorded by a harvester after felling (squares).

The figure shows that the scanner data is noisy (due, e.g., to nodal swelling, dead branch stubs, dead needles, partially hidden stem, etc.), and there are sections of the tree for which the scanner has no readings. Taper prediction involves interpolation of missing sections, extrapolation to the upper reaches of the tree, and smoothing of noise.

In fact, for each reading in a query, we also have a confidence measure. This is a measure of the confidence the TreeMetrics software has in the accuracy of the reading. The software uses 3D image recognition techniques: the 3D coordinates recorded by the scanner are assigned to different trees. The height at which a reading is taken can be determined to a high level of accuracy, but diameters are less reliable due, e.g., to occlusion by parts of neighbouring trees. (There is no equivalent to these confidence measures in cases because, as we described above,

cases describe trees that have been properly measured, either manually or by a harvester. We assume that all readings in cases are ones we can be confident in, although this can be achieved only under carefully controlled circumstances.)

Formally, a case  $c$  is a set of pairs,  $c = \{\langle i_1, x_1 \rangle, \dots, \langle i_m, x_m \rangle\}$ , where  $x_1, \dots, x_m$  are measurements, usually stem diameters, and  $i_1, \dots, i_m$  are the heights at which the measurements were made, e.g.  $\langle 5, 300 \rangle$  means that at a height of 5 m from the base, the tree's diameter is 300 mm. For each case, it is expected that  $i_1, \dots, i_m$  will be consecutive heights.

A query  $q$  is a set of triples,  $q = \{\langle i_1, x_1, w_1 \rangle, \dots, \langle i_n, x_n, w_n \rangle\}$ , where  $x_1, \dots, x_n$  are stem diameters calculated from scanner readings,  $i_1, \dots, i_n$  are the heights at which the readings were taken, and  $w_1, \dots, w_n$  measure confidence in the readings. For queries, it is typically not the case that heights  $i_1, \dots, i_n$  are consecutive.

### 3.2 Similarity

We need to be able to compute the similarity between cases and queries. In fact, we define a distance function, rather than a similarity measure, whose design is informed in part by the following observations about cases (usually harvester data) and queries (scanner data):

**Sequence data:** Cases and queries both have the characteristics of sequence data. Each data point  $x_j$  has a definite relationship with those either side of it,  $x_{j-1}$  and  $x_{j+1}$ . The data is effectively the description of a shape. There is an analogy here between our cases and temporal cases, where values are recorded at different points in time.

**Varying lengths:** Stems vary in height and the raw series data reflect this. Since measurements and readings are taken at fixed intervals, the length of a sequence varies from stem to stem. Cases need not be the same length; queries need not be the same length; and queries need not be the same length as cases. This means that any vector-based similarity measures that assume fixed-length vectors cannot be used directly.

**Partially incomplete:** Both the cases (harvester data) and queries (scanner data) can be incomplete sequences, in the sense that there may not be measurements or readings at certain heights. However, they are incomplete in distinct ways. The harvester data will have a measurement at every interval up to a certain height but may not completely record the final taper of the stem. The point at which the sequence terminates varies with each individual file. The scanner data in contrast contains many missing sections of data due to occlusion from branches and other trees. These effects become more prominent further up the stem. As a result there is typically more information about the sequence at the base of the tree; readings are fewer and more sparsely distributed further up the stem.

We investigated a number of variations of an existing shape-based similarity measure [21], but without great success. We believe that this is because this

measure, the variants we tried, and others like it tend to assume that cases and queries are quite homogeneous and symmetric. In our forestry system, however, we have seen that cases and queries are quite different from each other.

For this reason, we have defined ASES, our own asymmetrical sequence-based Euclidean distance function, which we believe is well-suited to the task at hand:

$$ASES(q, c) = \sqrt{\frac{\sum_{\langle i, x, w \rangle \in q} w \times diff(i, x, c)}{\sum_{\langle i, x, w \rangle \in q} w}} \quad (1)$$

where

$$diff(i, x, c) = \begin{cases} x^2 & \text{if } i \notin \{i' \mid \langle i', x' \rangle \in c\} \\ (x - x')^2 \text{ such that } \langle i, x' \rangle \in c & \text{otherwise} \end{cases} \quad (2)$$

In essence, this global distance measure is a weighted sum of local distances, where the weights are the confidence measures from the query. For each reading in the query, a local distance is computed. If the query contains a reading  $\langle i, x, w \rangle$  and the case contains a measurement that was taken at the same height  $i$  (i.e. if it is the case that  $\langle i, x' \rangle \in c$ ), then the local distance is the square of the difference between the query reading and the case measurement,  $(x - x')^2$ . If the case does not contain any measurement taken at height  $i$  (i.e.  $i \notin \{i' \mid \langle i', x' \rangle \in c\}$ ), then the local distance is the square of the whole amount of the reading,  $x^2$ . This has the effect of penalizing cases that are too short to match all the readings in the query.

### 3.3 Retrieval and Reuse

Of the four phases in the CBR cycle the two most critical in Cabar are the retrieval and reuse phases. At present, we have experimental results only for quite simple versions of these phases. We have tried more sophisticated techniques, but we will not describe them here because they have not been verified experimentally. We explain how we carry out tree taper prediction, stem volume prediction, and product breakout estimation.

**Tree Taper Prediction.** Given a query  $q$ , we retrieve its  $k$  nearest neighbours using the ASES distance measure. In the simple approach for which we have experimental results, we use only  $k = 1$ . We use this nearest neighbour  $c$  for query completion. In fact, the only approach for which we have experimental results is the very simplest one: we take the whole of  $c$  unchanged (i.e. without adaptation) in place of  $q$ .

**Stem Volume Prediction.** Stem volume prediction is trivial once the query has been completed. It involves no more than computing the volume based on the inferred diameters.

**Product Breakout Estimation.** Product breakout estimation is the most complex task that we carry out. Basically this involves giving a prospective processing plant such as a sawmill a sense of the likely products that can be produced from the stems captured by the scanning device.

As explained in Section 2.1, given a specification of the products that a processing plant is interested in, we can use a cross-cutting algorithm to simulate the actions that a harvester machine would carry out in the forest. Such cross-cutting algorithms are commonly used in forestry and we have designed and developed an adaptation of one which can utilize the extra 3D information captured by the TreeMetrics scanning device. The extra information gives the algorithm knowledge of tree curvature. Our algorithm therefore gives more accurate estimates of the breakout because it better takes problems of curvature into account.

The cross-cutting algorithm we use is an adaptation of the branch-and-bound approach proposed by Bobrowski, which is proven to produce the optimal solution [12]. As each possible solution path in its search tree is evaluated, our variant of Bobrowski's algorithm ensures that any restrictions the products place on acceptable curvature levels are taken into account.

## 4 An Experimental Evaluation of Cabar

Although being able to examine Cabar in real world situations is the ultimate proof of concept, such studies often contain multiple sources of error. Such errors, especially when unquantifiable, make it difficult to properly assess the performance of a new technology and impossible to isolate different areas of failure. In this paper we wish to examine and quantify the performance of Cabar over a range of different scenarios, in particular where there are varying quantities of data and varying levels of noise. To achieve this we developed a testbed to simulate these situations.

The benchmark against which we compare Cabar's performance is similar to many used in forestry and much the same as those described in Section 2.1. It predicts tree taper and then stem volume using a taper equation whose parameter is a DBH measurement, and it estimates product breakout using a harvester simulation model.

We first describe our experimental data; then we describe our benchmark system; finally we present the results of our experiments.

### 4.1 Experimental Data

In our experiments we use harvester data, which we assume to be accurate. Obviously, this is the source of our case data. But it is also the source of our query data. We applied ablation and noise functions to harvester data in order to simulate scanner data of varying quality. The reason we create queries from harvester data is that we need to have a known 'ground truth' against which we can compare Cabar's predictions. What we need in future are readings produced by laser scanning a stand of trees paired with harvester measurements on the

same trees after felling. Lacking enough data of this kind, we were obliged to use harvester data alone. We expect this to be remedied in the future.

In particular, the harvester files used in our experiments are Sitka Spruce tree files from Ireland. We removed from this set of harvester files any which did not have a complete set of readings up to the 70 mm diameter point and also any files in which substantial discontinuities occurred. This left a set of 389 tree stems. We then split the remaining data into two separate, equal-sized data sets. One set was used to generate queries of simulated scanner data; the other formed a case base. We applied ablation and noise functions to the measurements in the query data in order to simulate scanner data, as described in detail below. In our experiments, confidence levels in the query data are all set to 1.

## 4.2 Ablation and Noise Functions

To decide whether to retain or delete a measurement  $\langle i, x \rangle$  in a query, we sample a random distribution. If a sampled value  $r$  is greater than probability of retention  $P(\alpha, \beta, i)$ , then  $\langle i, x \rangle$  is retained, and otherwise it is deleted. Retention/ablation probabilities are based on the generalized exponential distribution [22]:

$$P(\alpha, \beta, i) = e^{-\frac{(i-\alpha)}{\beta}} \quad (3)$$

$\alpha$  defines the shape of the distribution, and  $\beta$  is a scaling factor. We can adjust the extent to which the readings that are higher in the tree are retained by changing the value of  $\beta$ . For the remainder of this document we will refer to  $\beta$  as the ablation factor.

Figure 2 shows an example harvester tree stem to which we have applied our ablation function. The points remaining after ablation are shown as squares. In this figure the effects of adding noise can also be seen (stars). The noise we added was normally distributed with a variance set to be a percentage of the original diameter. Our Treemetrics expert informally confirmed that this simulated scanner data strongly resembles real scanner data.

We applied varying degrees of ablation and noise to the queries. We altered the noise levels from no noise at all up to a noise level of 20% [3]. The ablation factor took on the values 1, 1.5 and 2.

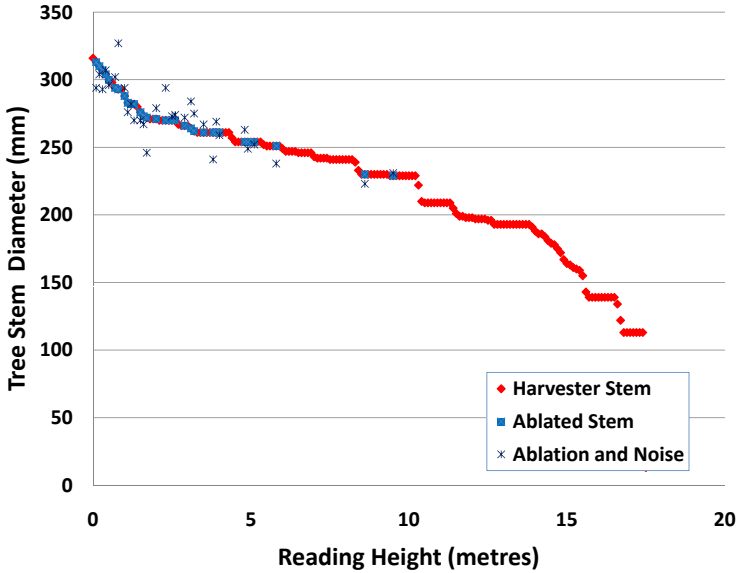
## 4.3 The Malone Kozak Benchmark System

In order to benchmark the performance of the CBR system against a realistic alternative, we developed a system similar to one of the approaches described in Section 2.1. One important element of this estimate mechanism is the taper equation used. We used a taper equation called the Malone Kozak. The Malone Kozak has been especially calibrated for Sitka Spruce in Ireland [10]. This taper

---

<sup>3</sup> Noise levels of up to 20% seemed reasonable at the time these experiments were run. Data we have recently acquired for South Australian stems shows that we may need in future to use a slightly larger variance.





**Fig. 2.** A example of a query and the harvester stem data from which it was created

equation uses DBH and height measurements as its inputs. But since height measurements are often not available, the Malone Kozak comes with a height model that predicts height based on DBH. Since our data-sets did not give us height measurements, we used the height model in our experiments. In generating queries from harvester data (above), we were careful to ensure that we did not ablate the DBH of each of the query trees and that we did not apply noise to the DBH. The Malone Kozak would be particularly susceptible to such noise and would be unusable without a DBH measurement.

For product breakout estimation, the benchmark system uses the same cross-cutting algorithm as the one we developed for Cabar (Section 3.3).

#### 4.4 Experimental Methodology

We took the queries and applied a particular level of ablation and noise. We then used Cabar to predict tree taper, stem volume, and product breakout market value. We repeated this 20 times using the same noise and ablation factors, and averaged the results. We then changed the noise and ablation factors and repeated this process for each different level of noise and ablation.

The Malone Kozak system, on the other hand, was run only once on the query set. Only one run is needed because we ensure that the noise and ablation factors do not alter the true DBH, which is the only input in the Malone Kozak equations that we use (Section 4.3).

**Table 1.** A description of the products harvested from an Irish forest

Product	Length (m)	SED (mm)	Market Value
4.9 Saw log	4.9	200	39
Pallet	3.1	140	25
Stake	1.6	70	19

To perform product breakout estimation, we need an example of product demand. For this, we chose a small set of products which are typical of those harvested in Ireland. A description of these products can be seen in Table 1. The single most important feature of each product is its length. However, each product is further described by other characteristics that restrict whether certain sections of a given tree can produce such lengths. We use the small end diameter (SED), which describes the minimum diameter that the upper or smaller diameter of a cut section is allowed to be. These restrictions are taken into account by our cross-cutting algorithm.

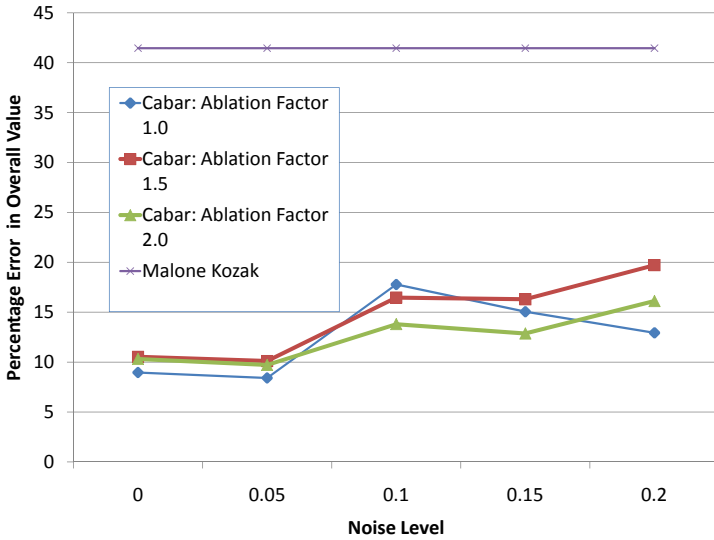
The cross-cutting algorithm also needs priorities, denoting the importance of each type of product. For our experiments, we used indicative market values as the set of weights describing the priorities.

The final outcome of the cross-cutting algorithm is the estimated overall value of the whole query set using the market values in Table 1. We compare these estimates from both systems to the ‘ground truth’, i.e. the market value of the products that can be cross-cut from the original, noise-free, unablated query tree data. We report the percentage error.

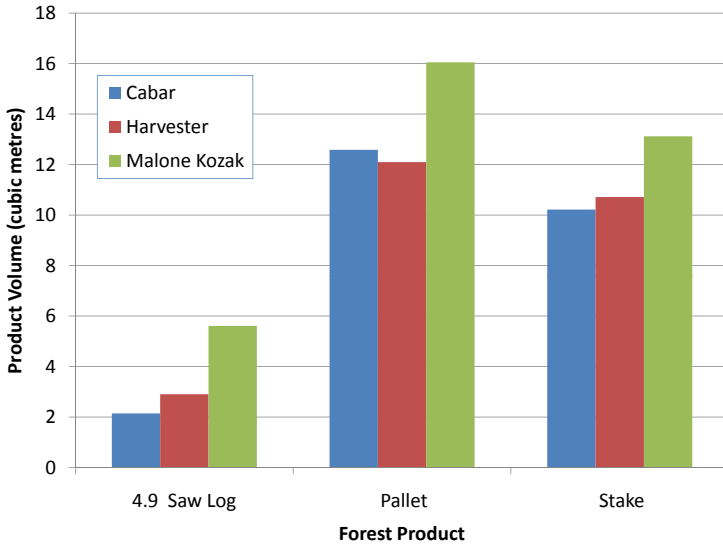
## 4.5 Results

Figure 3 shows the percentage error in the estimates of total product breakout market values for the query set. Two systems are compared: Cabar and Malone Kozak (with a generic height model). In the case of Cabar, there is a separate trend line for each of the three ablation factors that we used. The graph plots the error against different levels of noise. The Malone Kozak system is insensitive to this, as explained in Section 4.4, because we use only the DBH and we ensured that this was noise-free and never ablated. It can be seen that Cabar’s error levels are far below the Malone Kozak levels. Of course, if the Malone Kozak system had been calibrated not just for Sitka Spruce in Ireland, but for the particular stand of trees used in the data-set, then its performance would be more competitive. This reinforces the point about the importance (and cost) of model calibration in approaches like this one. Similarly, supplying tree height readings if they had been available instead of using the height model would have made the Malone Kozak system more competitive.

The source of Cabar’s greater accuracy becomes clearer when we look at the product breakout predictions for particular noise and ablation factors. Figure 4, for example, shows the product breakout volumes by product type when the noise and ablation values are 0.15 and 1.5 respectively. Cabar’s predictions are



**Fig. 3.** Product breakout market value error results



**Fig. 4.** Product breakout volumes (noise factor 0.15; ablation factor 1.5)

generally reasonably close to the figures produced by the cross-cutting algorithm on the actual stems (first two bars in each group of three). Malone Kozak, by contrast, tends to greatly overestimate the sizes of the trees and hence the product breakout volumes (third bar in each group). This indicates that the trees

used in the harvester data-set were shorter and had greater taper than predicted by the Malone Kozak equations and height model.

## 5 Conclusions and Future Work

Improvements in terrestrial scanner technology mean that it is now possible to collect far more information about a forest in advance of it being harvested. However, this data does not offer complete profiles of the trees in the forest. Systems that can ‘fill in the gaps’ are needed. Traditional approaches to these estimation tasks were not designed with such rich data sources in mind and are unable to exploit them. In this paper we presented Cabar, a Case-Based Reasoning approach, which is better suited to dealing with the abundance of scanner data but also the challenges such data poses. Our preliminary results demonstrate that Cabar provides a viable alternative solution to some of the challenges currently hindering the adoption of terrestrial scanning in forestry.

Although the work we have presented demonstrates that a CBR approach to this problem is promising, there are several possible ways in which it might be improved. The first step would be to extend our system to operate on the  $k$  nearest neighbours for  $k > 1$  and to develop a more sophisticated approach to case completion from the nearest neighbours. We are also contemplating alternative distance functions that take account of area rather than diameter, which would tend to give greater weight to differences nearer the base of the tree. We are also considering how to make more use of the curvature data that the scanning technology gives us. We use it in our cross-cutting algorithm, but it is not used in the distance function. Using it in the distance function raises methodological problems because no harvester measures this attribute, hence ‘ground truth’ figures would not be readily available. More empirical evaluation is also called for. We are collecting further data sets on which experiments can be run. In particular, it is likely that we will have data that contains scanner readings and harvester measurements for the same trees. With this, we will not need to use simulated queries, and we can investigate the role of the confidence measures.

## References

1. Boston, K., Murphy, G.: Value recovery from two mechanized bucking operations in the Southeastern United States. *Southern Journal of Applied Forestry* 27(4), 259–263 (2003)
2. Murphy, G.: Mechanization and value recovery: worldwide experiences. In: *Forest Engineering Conference: Forest Engineering Solutions for Achieving Sustainable Forest Resource Management – An International Perspective*, pp. 23–32 (2002)
3. Kivinen, V.P.: Design and testing of stand-specific bucking instructions for use on modern cut-to-length harvesters. PhD thesis, University of Helsinki (2007)
4. Marshall, H.D.: An Investigation of Factors Affecting the Optimal Log Output Distribution from Mechanical Harvesting and Processing Systems. PhD thesis, Oregon State University (2005)

5. Gordon, A., Wakelin, S., Threadgill, J.: Using measured and modelled wood quality information to optimise harvest scheduling and log allocation decisions. *New Zealand Journal of Forestry Science* 36(2/3), 198–215 (2006)
6. Newnham, R.M.: Variable-form taper functions for Alberta tree species. *Canadian Journal of Forest Research* 22, 210–223 (1992)
7. Lappi, J.: A multivariate, nonparametric stem-curve prediction method. *Canadian Journal of Forest Research* 36(4), 1017–1027 (2006)
8. Uusitalo, J.: Pre-harvest measurement of pine stands for sawlog production planning. Department of Forest Resource Management Publications, University of Helsinki, Finland (1995)
9. Curtis, R.O.: Height-diameter and height-diameter-age equations for second growth Douglas fir. *Forest Science* 13(4), 365–375 (1967)
10. Nieuwenhuis, M.: The development and validation of pre-harvest inventory methodologies for timber procurement in Ireland. *Silva Fennica* 36(2), 535–547 (2002)
11. Nasberg, M.: Mathematical programming models for optimal log bucking. PhD thesis, Linköping University (1985)
12. Bobrowski, P.M.: Branch-and-bound strategies for the log bucking problem. *Decision Sciences* 21(4), 1–13 (1990)
13. Sessions, J., Layton, R., Guangda, L.: Improving tree bucking decisions: a network approach. *The Compiler* 6(1), 5–9 (1988)
14. Bienert, A., Scheller, S., Keane, E., Mohan, F., Nugent, C.: Tree detection and diameter estimations by analysis of forest terrestrial laserscanner point clouds. In: *ISPRS Workshop on Laser Scanning 2007*, pp. 50–55 (2007)
15. Bienert, A., Scheller, A., Keane, E., Mulloly, G., Mohan, F.: Application of terrestrial laser scanners for the determination of forest inventory parameters. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36(5) (2006)
16. Goodenough, D.G., Charlebois, D., Bhogal, A.S., Matwin, S., Daley, N.: Automated forestry inventory update with SEIDAM. In: *Procs. of the IEEE Geoscience and Remote Sensing Symposium*, pp. 670–673 (1997)
17. Kelly, M., Cunningham, P.: Building competent compact case-bases: A case study. In: *Procs. of the Ninth Irish Conference in Artificial Intelligence & Cognitive Science*, pp. 177–185 (1998)
18. Amishev, D., Murphy, G.: Implementing resonance-based acoustic technology on mechanical harvesters/processors for real-time wood stiffness assessment: Opportunities and considerations. *International Journal of Forest Engineering* 19(2), 49–57 (2008)
19. Nummi, T.: Prediction of stem characteristics for *pinus sylvestris*. *Scandinavian Journal of Forest Research* 14, 270–275 (1999)
20. Liski, E., Nummi, T.: Prediction of tree stems to improve efficiency in automatized harvesting of forests. *Scandinavian Journal of Statistics* 22, 255–269 (1995)
21. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision* 40(2), 99–121 (2000)
22. Balakrishnan, N., Basu, A.P.: *The Exponential Distribution: Theory, Methods, and Applications*. Gordon and Breach, New York (1996)

# A Case-Based Perspective on Social Web Search\*

Barry Smyth, Peter Briggs, Maurice Coyle, and Michael P. O'Mahony

CLARITY: Centre for Sensor Web Technologies  
HeyStaks Ltd., NovaUCD, University College Dublin, Ireland  
<http://www.clarity-centre.org>, <http://www.heystaks.com>

**Abstract.** Web search is the main way for millions of users to access information every day, but we continue to struggle when it comes to finding the right information at the right time. In this paper we build on recent work to describe and evaluate a new application of case-based Web search, one that focuses on how experience reuse can support collaboration among searchers. Special emphasis is placed on the development of a case-based system that is compatible with existing search engines. We also describe the results of a live-user deployment.

## 1 Introduction

Mainstream search engines like Google are the primary way that people access information online, but their success is muted by the challenges that remain. Even Google fails to deliver relevant results as much as 50% of time [1]. This leads to poor search productivity but is as much a result of the vague queries that are commonplace in Web search (e.g. [2]) as it is due to failings in core search engine technology. Unfortunately vague queries are the reality of Web search and so in response there has been considerable research on different ways to improve result selection and ranking. For example, researchers have looked at ways to add context to bias search in the direction of special types of information (e.g., people, papers, etc.); see for e.g. [3]. Others have attempted to profile the preferences of searchers to deliver more personalized result-rankings [4,5,6].

Case-based reasoning (CBR) researchers have also recognised the opportunity for case-based techniques to improve Web search and information retrieval. For example, the work of Rissland [7] looks at the application of CBR to legal information retrieval, and [8] describe a case-based approach to question-answering tasks. Similarly, in recent years there has been considerable research looking at how CBR techniques can deal with less structured textual cases. This has led to a range of so-called *textual CBR* techniques [9]. In the context of Web search, one particularly relevant piece of work concerns the *Broadway* recommender system [10], and specifically the Broadway-QR query refinement technique that uses case-based techniques to reuse past query refinements in order to recommend new refinements to searchers. The work of [11] apply CBR techniques to Web search in a different way, by combining user profiling and textual case-based reasoning to dynamically filter Web documents according to a user's learned preferences.

---

\* This work is supported by Science Foundation Ireland under grant 07/CE/I1147.

This paper also focuses on how CBR techniques can be applied to Web search. It builds on previous work [12,13,14] which has already demonstrated the benefits of reusing search experiences within community-based search case bases; each case base representing the prior search experiences of a community of like-minded searchers. The main contribution of this paper is not a new case-based Web search technique per se – in fact we base our core CBR engine on the work of [13] – but rather a novel application of CBR techniques, which has the potential to contribute to mainstream Web search in a practical way. In the next section we describe the HeyStaks ([www.heystaks.com](http://www.heystaks.com)) Web search utility, which is designed to work alongside search engines such as Google. HeyStaks allows searchers to create case bases to better organise their search experiences. These can be shared with friends and colleagues and used to augment Google’s own search results with recommendations based on the search experiences of others. In Section 3 we describe a detailed evaluation of a recent HeyStaks deployment to demonstrate the value that HeyStaks brings to Web search; we show that users create and share search experiences at will, and benefit frequently from the experiences of others in a manner that mainstream search engines simply cannot support. In short we argue that this implementation of case-based Web search adds a new layer of collaboration to Web search by mediating the reuse of search experiences.

## 2 HeyStaks

HeyStaks is a new approach to case-based Web search, designed to work in cooperation *with* rather than competing *against* mainstream search engines. We describe HeyStaks as a *search utility* that seamlessly integrates with leading search engines such as Google, via a browser toolbar/plugin, to offer a number of practical features that are missing from today’s Web search engines.

### 2.1 A Motivating Example

To understand the motivation for HeyStaks, consider a common use-case: Stella is a new machine learning PhD student. Her early work is dominated by background research and she spends a lot of time searching for relevant material and papers on a range of research related topics. As a newcomer Stella often struggles to find the right queries and successful searches can be elusive. She is not a dedicated bookmarker and so often wastes time re-searching for information she has found previously; recent research suggests that up to 30% of our searches are about re-finding information that we have previously found; see [15].

Fortunately for Stella, she has joined a mature, cooperative research group and she benefits from the generous wisdom of her colleagues when it comes to better understanding the things she should be searching for. In the spirit of collaboration colleagues will often email Stella links to Web sites and papers that they have found. Once again, research supports the value of this type of ‘search’ collaboration: up to 70% of the time we find that colleagues and friends have already found the type of things that we are looking for online; see [15].

Stella’s experience is common: whether it is a group of colleagues at work, family members planning a vacation, or a set of friends researching an event, collaboration is frequently based on shared search experiences. Yet mainstream search engines offer no support for this type of collaboration, nor do they help individuals to organise their own search experiences. We believe that CBR techniques can play a key role in harnessing this type of collaboration in Web search.

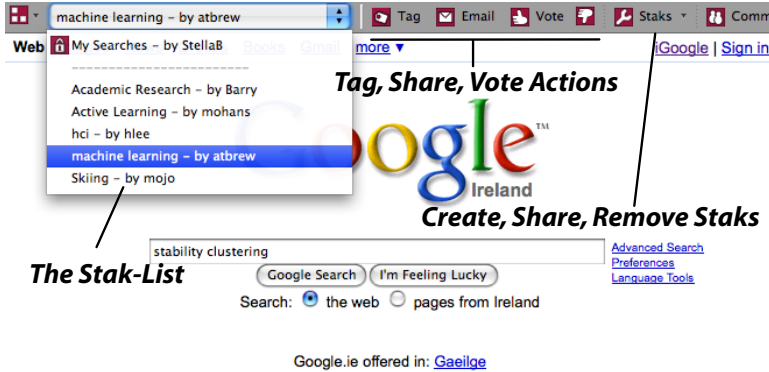


Fig. 1. Selecting an active search stak at the start of a new search

HeyStaks adds these missing features (the ability to *organise* and *share* search experiences) to Web search, via the HeyStaks browser toolbar. This allows Stella to group her searches into *staks* to reflect her different search interests (specific projects or tasks at work, travel, events etc.); these staks are case bases of search experiences. In the future, when Stella searches — her query defining a new information need — HeyStaks recommends results that she may have selected for similar queries. These reminders appear as promotions within the standard Google result-list. Moreover, Stella can chose to *share* her search staks with others, or she can join the search staks that others have created. Shared staks combine the search experiences of many, so that members can benefit from promotions that are derived from the experiences of others.

For example, in Fig. 1 we see that Stella has joined staks created by other group members. She selects the “Machine Learning” stak as she embarks on a search for “stability clustering” and the result-list is shown in Fig. 2. The top 3 results are HeyStaks promotions, recommended from stak activity for similar queries; these promotions may have been promoted from much lower down the Google result-list, depending on stak activity. Google’s normal results remain intact. At the top of the result-list is a reminder that HeyStaks has found additional recommendations. These come from Stella’s own “My Searches” stak, as well as the “Machine Learning” stak, but are not quite relevant enough to merit an automatic promotion. By expanding this list Stella can see these additional promotions. In this way HeyStaks provides a very powerful collaboration platform for Web search: users search with their favourite search engine as normal;



they organise their searches in to meaningful staks; and they benefit from the search experiences of their friends and colleagues.

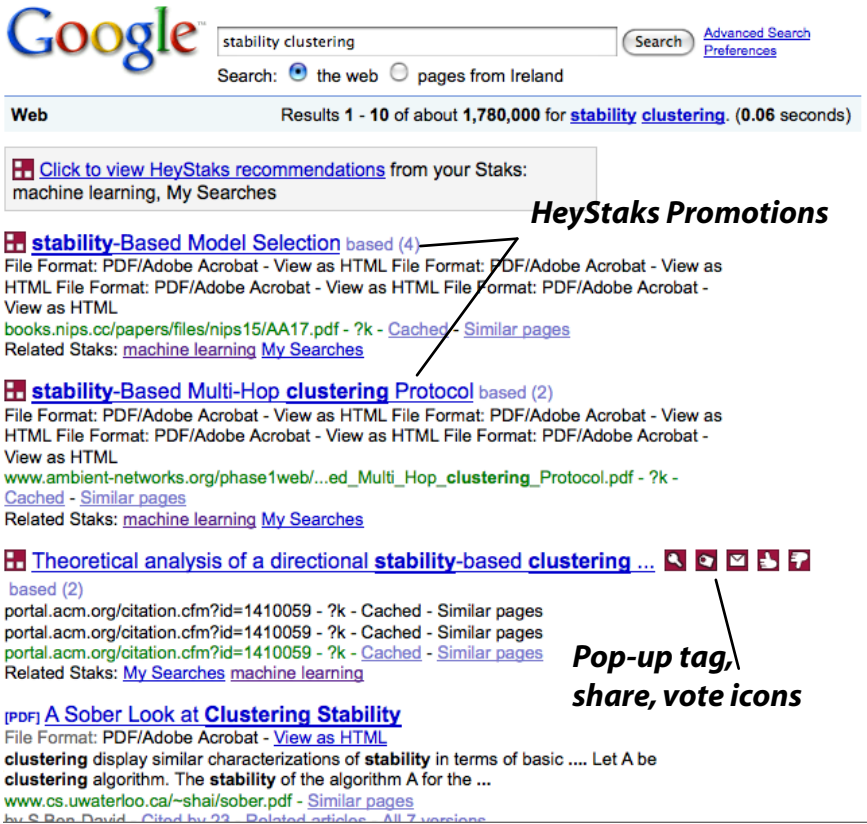


Fig. 2. Google search results with HeyStaks promotions

## 2.2 System Overview

Fig. 3 outlines the basic HeyStaks architecture. The system is made up of a number of key components including:

1. The HeyStaks *Server* is responsible for managing the core HeyStaks data including the user and stak databases (DB), which contain all of the basic user and stak information, and the search staks or case bases (CB), which store the individual stak-based search experiences.
2. The *Recommendation Engine* performs a number of recommendation tasks, the primary one being the selection of suitable results for promotion to the user, given the current target query and active stak. In addition the recommendation engine is also used to proactively suggest appropriate staks to

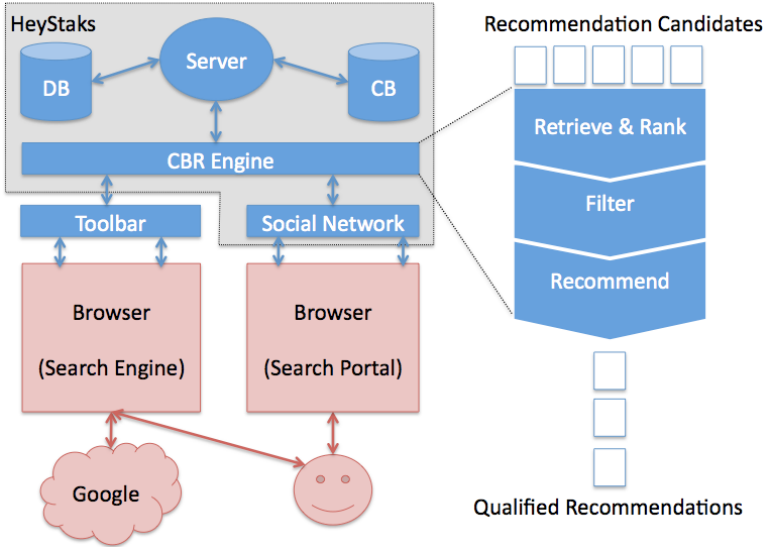


Fig. 3. The HeyStaks system architecture and outline recommendation model

Fig. 4. The HeyStaks search portal provides direct access to staks and past searches

searchers, according to their search context, but we will not focus on this feature in this paper.

3. The HeyStaks *Social Network* provides users with access to their individual profiles, the staks that they create and share, plus stak-related activity feeds as a way to keep up to date with what friends and colleagues have been searching for. It also provides a range of features to help users to maintain their staks and to help them discover interesting results and staks; for example Fig. 4 shows a stak page from the HeyStaks search portal for the “Machine Learning” stak discussed previously.
4. The HeyStaks *Toolbar* is the key client-side component. It provides users with various features to create and share staks and pages, and also acts as a link between the underlying search engine (e.g. Google) and the HeyStaks server, recording search experiences and inserting relevant promotions.

### 2.3 Case Bases of Search Experiences

As a CBR system, HeyStaks is distinguished in a number of interesting ways. In conventional CBR systems there is often a single case base or, at best, a small number of case bases, which have been created by trusted experts. HeyStaks is very different. Each stak is a case base and each user can create many staks. Moreover, there are few guarantees about the reliability of the information that makes its way in to these case bases. Users might forget to select an appropriate stak for their search, and although there is an automatic stak selection feature (beyond the scope of this work) it is not 100% reliable. These problems are exacerbated by the fact that a typical stak may be subscribed to by many different stak members, each with his/her own individual agenda for participating in the stak. All of this makes for a much more fluid and dynamic (and challenging) CBR environment, where experiences are distributed across a multitude of evolving case bases. It is, incidentally, also an environment in which understanding the source of case knowledge is likely to be particularly important, because understanding the provenance of a case, and the reputation of the case and/or stak creator, can play a vital role when it comes to evaluating the case’s reliability; see [16, 14].

These ideas and challenges make fertile ground for future work but for now we will focus on describing the basic case representation and recommendation techniques that are used in HeyStaks. Each search stak ( $S$ ) is a case base recording the search experiences of the stak members. Each stak is made up of a set of cases ( $S = \{c_1, \dots, c_k\}$ ) and each case corresponds to a single result page ( $p_i$ ) that has been ‘selected’ for this stak. Each case is anonymously associated with a number of implicit and explicit interest indicators, including: the total number of times the result has been selected ( $sel$ ) during a search, the query terms ( $q_1, \dots, q_n$ ) that led to its selection, the snippet terms associated with the result when it wasn’t selected ( $s_1, \dots, s_m$ ), the total number of times a result has been tagged ( $tag$ ) and the terms used to tag it ( $t_1, \dots, t_w$ ), the total votes it has received ( $v^+, v^-$ ), and the number of people it has been shared with ( $share$ ); see Eq. 1. In addition each term (query, tag, snippet) is linked with a hit-count

that reflects the number of times that this term has been associated with the page in question; for example,  $h_{q_1}$  refers to the number of times that the query term  $q_1$  has been found in queries that have led to the selection of the page  $p_i$ .

$$c_i^S = \{p_i, (q_1, h_{q_1}) \dots (s_1, h_{s_1}) \dots (t_1, h_{t_1}) \dots (t_w, h_{t_w}), v^+, v^-, sel, tag, share\} \quad (1)$$

In this way, each case is associated with *term data* (query, snippet, tag terms) and *usage data* (the selection, tag, share, and voting counts). The former provides the basis for retrieving and ranking *promotion candidates*, while the latter provides a source of evidence that can be used to filter results and to generate a final set of recommendations. Thus, at search time, a set of recommendations is produced in a number of stages: relevant results are retrieved and ranked from a set of suitable staks; these promotion candidates are filtered, based on an *evidence model*, to eliminate noisy recommendations; and finally they are added to the Google result-list according to a set of *recommendation rules*.

**Retrieval & Ranking.** For a given target query,  $q_t$ , HeyStaks generates a set of promotion candidates. Briefly, there are two types of promotion candidates: *primary promotions* are results that come from the active stak<sup>1</sup>  $S_t$ ; whereas *secondary promotions* come from other staks in the searcher's stak-list.

$$score(q_t, c_j) = \sum_{t \in q_t} tf(tec_j) \bullet idf(t)^2 \quad (2)$$

To generate these promotion candidates, the HeyStaks server uses  $q_t$  as a probe into each stak case base,  $S_i$ , to identify a set of relevant stak cases  $C(S_i, q_t)$ . Each candidate case,  $c_j$  is scored using a similar technique to that described by [13] by using a TFIDF (*term frequency* • *inverse document frequency*) function as the basis for an initial recommendation ranking; this approach prefers cases that match terms in the query which have occurred frequently in the case, but infrequently across the case base as a whole (see Eq. 2).

**Evidence-Based Filtering.** We have already mentioned that search staks are inevitably noisy because searchers will often forget to set an appropriate stak at the start of a new search session. As a result the retrieval and ranking stage may select misclassified pages that are not relevant to the current query context. To avoid making spurious recommendations HeyStaks uses an *evidence filter*, which uses a variety of threshold models to further evaluate the relevance of a particular result in terms of its usage evidence; for instance, tagging evidence is considered more important than voting, which in turn is more important than implicit selection evidence. The precise details of this model are beyond the scope of this paper but, for example, pages that have only been selected once

<sup>1</sup> That is, the stak that is currently active in the HeyStaks toolbar. This stak will either have been selected by the user during a recent search or may have been automatically selected by HeyStaks based on the current query/search context.

by a single stake member are not automatically considered for recommendation and, all other things being equal, will be filtered out at this stage. In turn, pages that have received a high proportion of negative votes will also be eliminated.

**Recommendation Rules.** The final task is to add the remaining *qualified recommendations* to the Google result-list. HeyStaks uses a number of different recommendation rules to determine when and where a promotion should be added. Once again, space restrictions prevent a detailed account of this component but, for example, the top 3 primary promotions are always added to the top of the Google result-list, and labelled using the HeyStaks promotion icons. If a remaining primary promotion is also in the default Google result-list then this is labeled in its default Google position. If there are still remaining primary promotions then these are added to the secondary promotion list, which is sorted according to page TFIDF scores. These recommendations are then added to the Google result-list as an optional, expandable list of recommendations, such as that shown at the top of Fig. 2.

### 3 Evaluation

As an application-paper our main aim in this work is to describe the HeyStaks application and, in particular, the results of a recent live-deployment obtained from a usage analysis of 95 HeyStaks beta users (excluding those users directly connected with the HeyStaks project) during the period October 2008 - January 2009. In this evaluation we will focus on two particular evaluation themes:

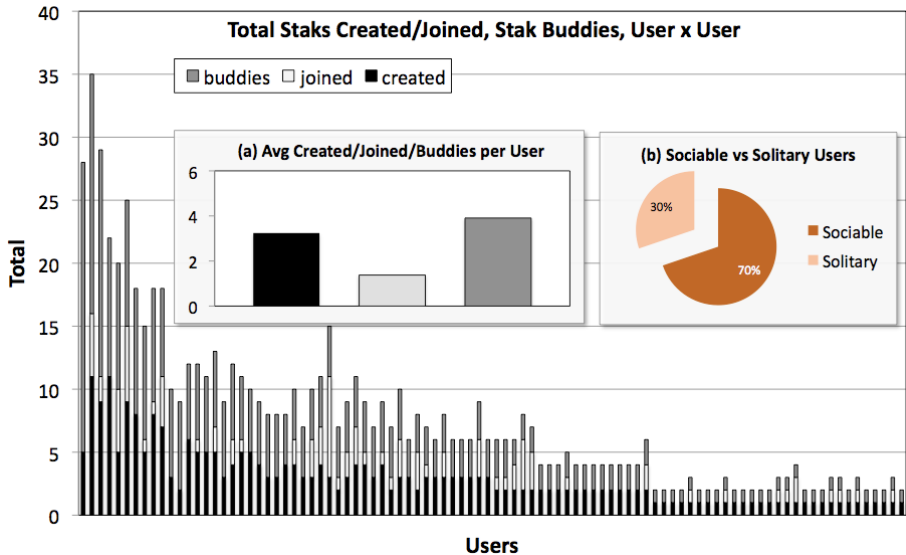
1. As a case-based reasoning system HeyStaks is interesting because it facilitates the casual creation and sharing of case knowledge, across multiple case bases, in the form of search experiences. But do users actually take the time to create these search case bases and do they share them with others?
2. As a search utility, HeyStaks is interesting because it promises to improve Web search by facilitating collaboration among searchers. But do users benefit from this collaboration? Do they respond positively to HeyStaks' promotions? Do they benefit from their own search experiences or those of others or a mixture of the two?

Since this is a study of live-users *in the wild* there are certain limitations on what we have been able to measure. There is no control group and it was not feasible, mainly for data privacy reasons, to analyse the relative click-through behaviour of users, by comparing their selections of Google results to their selections of promotions. However earlier work does report on these type of results in more conventional control-group laboratory studies (see for e.g. [15, 11]).

#### 3.1 On the Creation and Sharing of Search Case Bases

A key element of the HeyStaks value proposition is that searchers need a better way to organise and share their search experiences and, specifically, that the

ability to create and share search staks will provide them with these features. But do users actually take the time to create staks? Do they share them or join those created by others? As a user shares and joins staks they effectively create a *search network*, the members of which are the other users who co-share these staks. How do these search networks evolve as a result of stak sharing?



**Fig. 5.** The number of staks created and shared by users and the size of their respective search networks (buddy lists)

These questions are addressed by Figure 5 which plots the number of staks that are created and joined by the beta users; also shown are the number of stak members (buddies) per stak. The results demonstrate that users do actively engage in considerable stak creation activity and they do regularly join the staks created by others. For instance, the plot shows that the most prolific stak creator has created 11 new staks during the course of the beta trial, while other users join up to 8 staks that others have created. And this sharing of search experiences helps create search networks that extend to as many as 19 other users.

In fact, as per the Figure 5(a) insert we see that, on average, users create 3.2 staks and join another 1.4, to develop search networks that link them directly to about 4 other users. Perhaps this is not surprising: most users create a few staks and share them with a small network of colleagues or friends, at least initially. Importantly though, the vast majority of users (70%) do make the effort to share/join staks — these are the *sociable* users of insert Figure 5(b) — and for staks that are shared, they benefit from an average of 3.6 members each.

In total more than 300 staks were created during the beta, on a wide range of interests, from broad topics such as travel, research, entertainment, to more niche interests including archaeology, black and white photography, and biking.

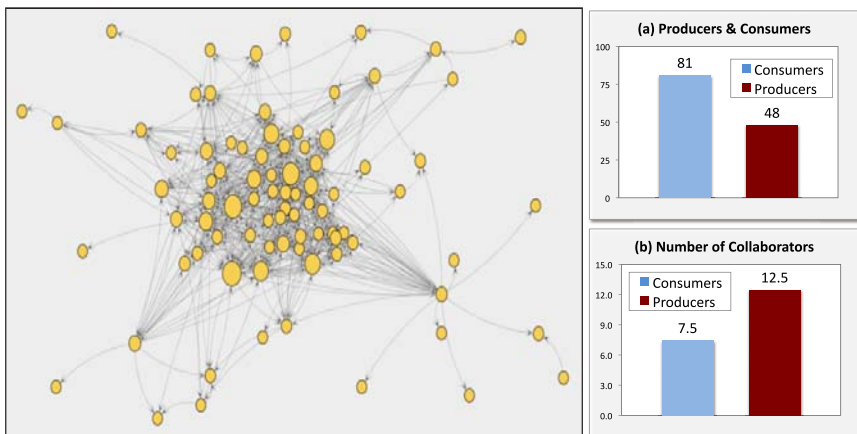
These data speak to the inherent willingness of users to take advantage of the organisation and sharing features that HeyStaks makes available, resulting in the creation of a large number of casual search case-bases that have the potential to evolve over time to become significant repositories of topical search knowledge.

Of course the creation and sharing of search staks is really just a means to an end: it is a way to help users partition their search interests to facilitate recommendation and recovery. Ultimately the success of HeyStaks will depend on whether users find these recommendations to be useful, which we will explore further in the following sections.

### 3.2 Search Collaboration

One of the most important observations about users is the extent to which their natural search activity creates a community of collaborating searchers. HeyStaks is motivated by the idea that Web search is an inherently social or collaborative activity, despite the absence of such collaboration features from mainstream search engines. In this section we will examine the nature of this collaboration effect, and the extent to which actual collaboration occurs in practice. As users search, tag, and vote they effectively produce and consume search knowledge. A user who is the first to select or tag a given result for a stak *produces* new search knowledge. Later, if this result is promoted to another user and re-selected, then this other user is said to have *consumed* that search knowledge.

These relationships between the producers and consumers of search knowledge within staks effectively creates an implicit social network of search collaboration. Fig. 6 presents a visualization of this network for the beta users. Each node is a unique user and edges between nodes correspond to evidence for search



**Fig. 6.** A representation of the collaboration network among HeyStaks searchers: (a) The percentage of searchers adopting consumer and producer roles; (b) The average number of collaborating searchers for producers and consumers



collaboration. These edges are directed: an edge from *user A* (the producer) to *user B* (the consumer) signifies that *user B* has selected at least one of the search results that *user A* has been responsible for adding (through his/her own selections, tagging or voting activity) to a search stack, which is shared between both users. Of course a single edge can (and typically does) reflect many collaboration instances between two users. In this example the diameter of the nodes reflects the *reputation* of the user in terms of their relative ability to help other users to search; however a detailed discussion of this reputation mechanism is beyond the scope of this paper.

Perhaps the first thing to notice is the extent of the collaboration that is evident among these users. From Fig. 6 we can see that the sharing of search knowledge is not limited to a small clique of especially social searchers, far from it: the graph includes 85% of beta users meaning that 85% of users have engaged in actual search collaborations. Most users (81 out of 95) have acted as consumers, benefiting from results that others have produced and 51% (48 out of 95) have acted as producers, creating at least some new search knowledge that others have consumed<sup>2</sup>; as shown in the Fig. 6(a) insert. Indeed when acting as a consumer, the average searcher benefits from results that have been produced by more than 7 other searchers. And when acting as a producer, searchers create search knowledge that is consumed by more than 12 other users on average; as shown in the Fig. 6(b) insert.

### 3.3 Search Leaders and Followers

These results tell us that searchers do appear to help others and that they are helped by others in return. In this section we take a more detailed look at the nature of search collaboration by examining the sources of promotions.

First let us start with an obvious next question by asking whether some users are especially altruistic in their searches, helping others more often than they are helped themselves? Figure 7 plots each user according to the number of other users they have helped (y-axis) compared to the number of other users they have been helped by (x-axis). As expected, given the data above, about half the users (those that are never producers) are situated on the x-axis<sup>3</sup>. These are the search *followers* within HeyStaks, the *novice* searchers who benefit disproportionately from the search experiences of others but are not seen to significantly contribute to search knowledge themselves. The other half of the user-base presents with a greater balance between the production and consumption of search knowledge, as evidenced by the clustering of data points along the main diagonal.

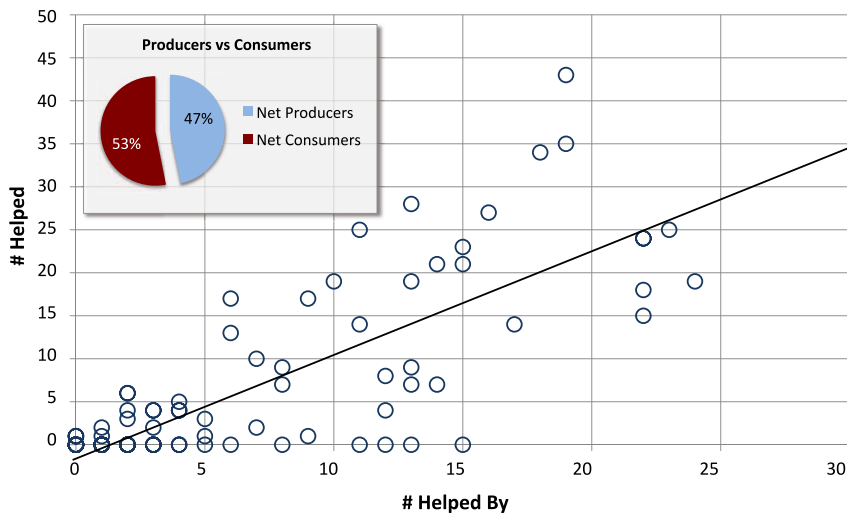
Users who have helped *more* people than they themselves have been helped by are *net producers*) and those who have been helped by more users than they have helped themselves, are *net consumers*). As the insert in Figure 7 shows, approximately 47% of users are net producers. This is especially important when we remember that above we noted how 51% of users have produced at least *some*

<sup>2</sup> Of course many users are both consumers and producers of search knowledge.

<sup>3</sup> Due to overlapping data points it is impossible to resolve each user along this axis.



search knowledge that has been consumed by some other user. The vast majority of *these* users, 94% of them in fact, are actually producing *more* search knowledge than they consume. These are the search *leaders* within HeyStaks.

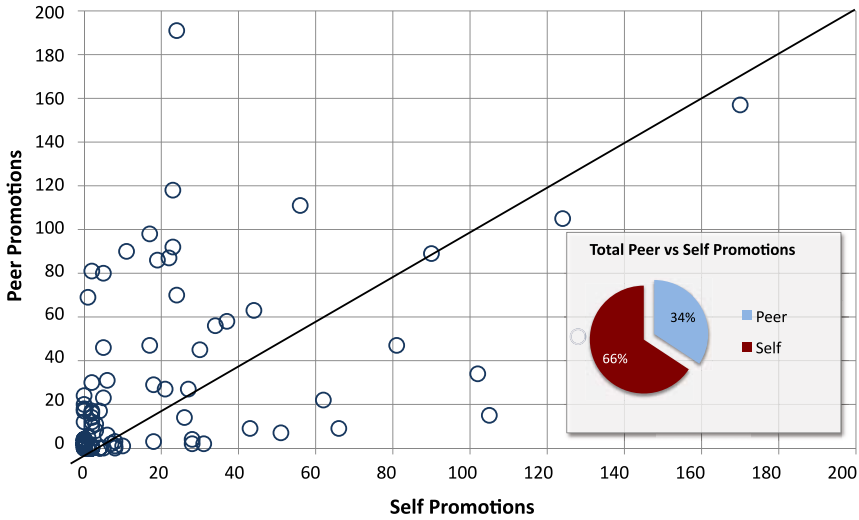


**Fig. 7.** Each HeyStaks user is plotted according to the number of other users that they have helped and been helped by. The points above the main diagonal indicate the net producers and those below are the net consumers.

### 3.4 Promotion Sources

So far we have looked at user-level collaboration and found that users do often collaborate through the implicit production and consumption of search knowledge. Now we will explore similar ideas but at the level of individual searches. Overall, the beta users selected more than 11,000 promotions as they searched. When a user performs a search and selects a promotion, how often is this promotion derived from their own past search activities (*self promotions*) and how often does it come from the shared search activities of other stak members (*peer promotions*)? The intuition here is that the selection of self promotions corresponds to examples of HeyStaks helping users to *recover* results they have previously found, whereas the selection of promotions from peers corresponds to *discovery* tasks (see [17]), where the user is benefiting from new content that might otherwise have been missed, or have been difficult to find.

Figure 8 presents this information as a plot of users located according to the actual number of peer and self promotions selected. For example, the somewhat isolated user represented by the data point in the top-left quadrant of the graph has selected about 25 self promotions but more than 190 peer promotions, clearly demonstrating the ‘discovery’ benefits of shared staks for this particular user. At the other extreme there are users (towards the bottom right quadrant) who benefit disproportionately from their own self promotions.



**Fig. 8.** Each HeyStaks user is plotted according to the number of peer and self promotions they have selected

As per Figure 8, 66% of all selected promotions are self promotions; users are usually benefiting from their own past searches. This is not surprising, especially during the early stages of stak development. Stak growth is initially led by the stak creator, and so inevitably most of the promotions that are generated are in response to the stak creator’s search queries. Most of these promotions will be self promotions, derived from the creator’s own search activities. As staks are shared, and more users join, the pool of searchers becomes more diverse: more results are added by the actions of peers and more peer promotions are inevitably generated and selected. It is an interesting task for future work to explore the evolution of a search stak and to further investigate how stak content and promotions are affected as more and more users participate. Are there well-defined stages in stak evolution, for example, as self promotions give way to peer promotions? For now it is satisfying to see that even in the early stages of stak evolution, where the average stak has between 3 and 4 members, that 34% of the time members are benefiting from promotions that are derived from the activities of their peers.

## 4 Conclusions

The social Web places emphasis on the sharing of experiences (e.g., comments, opinions, ratings) and CBR has a opportunity to play an important role; see [18]. This paper adds to a body of research on case-based Web search. The main contribution is a description of the HeyStaks application and a detailed evaluation of its beta launch. The results of this evaluation highlight the potential for CBR to add value in the social Web. We have highlighted how the HeyStaks

approach helps to support a very beneficial form of search collaboration that has been missing from mainstream search. In concluding, it is worth highlighting the CBR questions that have emerged during the course of this work:

1. We have focused on recommending cases from an individual case base (stak) but in HeyStaks there are many different case bases and the ability to suggest a case base is a useful feature in highlighting a new source of search knowledge to users. How might case bases be recommended?
2. Given that our search case bases are much noisier than expert-created case bases, how should this influence retrieval, reuse, and maintenance?
3. As cases are reused they are *enriched* by the activities of others: each case evolves to contain a *trace* of its reuse, as users re-select, tag, vote, and share the page associated with the case; see also [19].
4. As case bases evolve there may be opportunities to merge related case bases, or case bases may start to diverge as different contributors use them in different ways. How might these opportunities to merge or split case bases be recognised and handled?

These questions highlight new opportunities for research that will help to further strengthen the role of CBR in the social Web. In turn, HeyStaks presents many interesting challenges when it comes to addressing the type of scale that will be faced in a broader deployment context. For example, additional recommendation challenges exist when it comes to recommending pre-existing staks for users to join, based on a user's current search query or proactively, based on their previous queries. Moreover, the issue of malicious users, and their ability to influence stak promotions needs also to be addressed, perhaps by evaluating the reputation of users and allowing this to influence promotions.

## References

1. Coyle, M., Smyth, B.: Information recovery and discovery in collaborative web search. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECIR 2007. LNCS, vol. 4425, pp. 356–367. Springer, Heidelberg (2007)
2. Spink, A., Wolfram, D., Jansen, M.B.J., Saracevic, T.: Searching the web: the public and their queries. *Journal of the American Society for Information Science and Technology* 52(3), 226–234 (2001)
3. Lawrence, S.: Context in Web Search. *IEEE Data Engineering Bulletin* 23(3), 25–32 (2000)
4. Liu, F., Yu, C., Meng, W.: Personalized Web Search for Improving Retrieval Effectiveness. *IEEE Transactions on Knowledge and Data Engineering* 16(1), 28–40 (2004)
5. Micarelli, A., Gasparetti, F., Sciarrone, F., Gauch, S.: Personalized search on the world wide web. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) *Adaptive Web 2007*. LNCS, vol. 4321, pp. 195–230. Springer, Heidelberg (2007)
6. Teevan, J., Dumais, S.T., Horvitz, E.: Personalizing search via automated analysis of interests and activities. In: *SIGIR 2005: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 449–456. ACM Press, New York (2005)

7. Rissland, E.L., Daniels, J.J.: A hybrid CBR-IR Approach to Legal Information Retrieval. In: Proceedings of the 5th international conference on Artificial intelligence and law, pp. 52–61. ACM Press, New York (1995)
8. Burke, R., Hammond, K., Kulyukin, V., Tomuro, S.: Question Answering from Frequently Asked Question Files. *AI Magazine* 18(2), 57–66 (1997)
9. Lenz, M., Ashley, K.: AAAI Workshop on Textual Case-Based Reasoning, AAAI Technical Report WS-98-12 (1999)
10. Kanawati, R., Jaczynski, M., Trouse, B., Andreloi, J.-M.: Applying the Broadway Recommendation Computation Approach for Implementing a Query Refinement Service in the CBKB Meta-search Engine. In: Conférence Française sur le Raisonnement à Partir de Cas (RáPC 1999) (1999)
11. Godoy, D., Amandi, A.: PersonalSearcher: An Intelligent Agent for Searching Web Pages. In: Monard, M.C., Sichman, J.S. (eds.) SBIA 2000 and IBERAMIA 2000. LNCS, vol. 1952, pp. 43–52. Springer, Heidelberg (2000)
12. Balfe, E., Smyth, B.: Case-based collaborative web search. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 489–503. Springer, Heidelberg (2004)
13. Boydell, O., Smyth, B.: Enhancing case-based, collaborative web search. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 329–343. Springer, Heidelberg (2007)
14. Briggs, P., Smyth, B.: Provenance, trust, and sharing in peer-to-peer case-based web search. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 89–103. Springer, Heidelberg (2008)
15. Smyth, B., Balfe, E., Freyne, J., Briggs, P., Coyle, M., Boydell, O.: Exploiting query repetition and regularity in an adaptive community-based web search engine. *User Model. User-Adapt. Interact.* 14(5), 383–423 (2004)
16. Leake, D.B., Whitehead, M.: Case provenance: The value of remembering case sources. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 194–208. Springer, Heidelberg (2007)
17. O’Day, V.L., Jeffries, R.: Orienteering in an information landscape: how information seekers get from here to there. In: Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 1993), pp. 438–445. ACM Press, New York (1993)
18. Plaza, E.: Semantics and experience in the future web. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 44–58. Springer, Heidelberg (2008)
19. Lafflaquière, J., Settouti, L.S., Prié, Y., Mille, A.: Trace-based framework for experience management and engineering. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS, vol. 4251, pp. 1171–1178. Springer, Heidelberg (2006)

# Determining Root Causes of Drilling Problems by Combining Cases and General Knowledge

Samad Valipour Shokouhi<sup>1</sup>, Agnar Aamodt<sup>2</sup>, Pål Skalle<sup>1</sup>, and Frode Sørmo<sup>3</sup>

<sup>1</sup> Department of Petroleum Technology (IPT)

<sup>2</sup> Department of Computer and Information Science (IDI)  
Norwegian University of Science and Technology (NTNU)  
NO-7491, Trondheim, Norway

<sup>3</sup> Verdande Technology AS

Stiklestadveien 1- Trondheim, Norway

valipour@ntnu.no, agnar.aamodt@idt.ntnu.no,  
pal.skalle@ntnu.no, frode@verdandetechnology.com

**Abstract.** Oil well drilling is a complex process which frequently leads to operational problems. In order to deal with some of these problems, knowledge intensive case based reasoning (KiCBR) has clearly shown potential. An important problem in oil well drilling is hole cleaning, in which a high number of observed parameters and other features are involved. This paper presents how to determine the root causes of poor hole cleaning episodes by means of KiCBR. This will help the drilling crew to apply a proper remedy. The effect of general domain knowledge was demonstrated in a comparative study, in which improved results in terms of similarity assessment and explanation capability were achieved.

**Keywords:** Case-based, knowledge intensive, oil well drilling.

## 1 Introduction

Drilling of oil wells is an expensive offshore operation, costing typically 200 000 US\$ per day. Any loss of time caused by unwanted events is costly. During drilling all material drilled out need to be removed, i.e. transported to the surface, a process which is referred to as hole cleaning. Often some of the material remains in the well, and hole cleaning is still among the most important problems to deal with during drilling. It is also one of the most studied phenomena within the petroleum industry. Insufficient hole cleaning can in extreme cases lead to loss of the well or a part of it, i.e. stop of the drilling process and blocking of the hole. Due to the number of parameters influencing hole cleaning and the complex mechanisms involved, the phenomenon has not yet been fully understood [1].

Case-based reasoning (CBR) is an approach to problem solving and decision making where a new problem is solved by finding one or more similar previously solved problems, called cases, and re-using them in the new problem situation. Application-oriented research in the area of case based reasoning has moved mature research

results into practical applications. Skalle et al [2] employed case based reasoning to improve efficiency of oil well drilling. Their focus was on lost circulation, which means that some of the drilling fluid that always fills the gap between the drill string and the well wall gets lost into fractures in the geological formation. They built fifty cases on the basis of information from one North Sea operator. A general domain model was used to match non-identical features that were related in the model. Mendes et al [3] presented an application of CBR in offshore well design. The result of that work was a formalization of the methodology for planning of an oil well in a case-based reasoning context. They used fuzzy set theory for the matching of index features. Popa et al [4] presented an application of CBR for planning and execution of well interventions, in order to improve the decision-making process. Abdollahi et al [5] explained the applicability of CBR for diagnosis of well integrity problems in order to reduce the risk of uncontrolled release of formation fluids into the well.

In the above systems general knowledge has been used in the case retrieval process, for feature matching. None of the systems, or other CBR applications in this domain, have taken advantage of general knowledge in order to help identify a problem solution. In the study presented here a model-based method has been implemented as a complementary tool in order to determine the root cause of a hole cleaning problem. In addition, parts of the model are also used to enhance matching quality. An experiment has been undertaken to study the effect of the causal model combined with cases, in comparison with cases only.

The rest of the paper is structured as follows: In chapter 2 we explain the hole cleaning problem in some more detail, related to the functionality of our system. Chapter 3 explains the case structure and similarity methods. In chapter 4 results from the study of the effect of the causal model is reported. The types of input to the reasoning system, and their relationships with causes of hole cleaning problems are described in chapter 5. The last chapter summarizes and concludes the paper.

## 2 The Hole Cleaning Problem

A drilling process consists of many steps, of which the actual drilling into the geological formation and the continuous cleaning of the borehole are core subprocesses. Fig. 1 illustrates the process at an abstract level. The hole cleaning issues arise when the drilling direction moves from vertical to deviated and horizontal hole angles. Horizontal drilling is getting more and more common, due to the increasing distance from the rig to productive wells. (“All the easy wells are already drilled”, as the phrase goes). Accumulation of solids at a given depth is a common source of pack off, which is a serious situation indicated by the building up of material inside the hole wall, with reduced hole diameter as a result.

Many studies have been carried out by other researchers related to the cleaning of deviated and horizontal holes [6], [7], [8], [9], [10], [11]. However, the results of the studies have so far not provided clear operational recommendations. One reason may be that such studies are focused on the role and effect of individual parameters. A CBR approach, on the other hand, allows us to view a larger set of parameters as a unit, without assuming particular restrictions on the parameters, such as parameter independence.

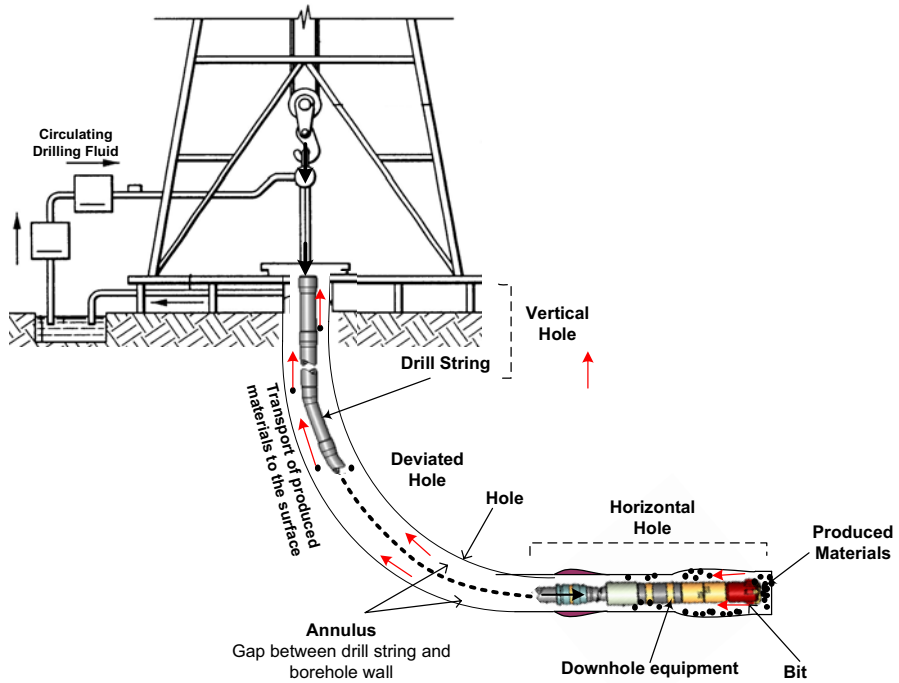


Fig. 1. Schematic drawing of an oil well being drilled

Our application is targeted at reducing the risk of unwanted downtime (i.e. stopped drilling). The drill plan acts as guidance to expected drilling behavior. The real-time data from the drilling process is the main source of a situation description, which is matched with a past case in order to identify possible hole cleaning problems ahead of the drill bit. When a sufficiently similar past case is found, the root cause for that problem is presented to the user. In our KiCBR approach this is supported by the causal model, linking input features to root causes.

### 3 Knowledge Assessment

#### 3.1 KICBR

A KiCBR system achieves its reasoning power through the set of previous cases combined with some other source of knowledge about a certain domain. Our system is designed for finding the root cause of a hole cleaning problem based on either the case base or the general knowledge module alone, or both of them in combination. To build the system, three knowledge models are needed:

- *A taxonomy:* extracting important terms from the domain.
- *A causal model:* building a model that describes causes and effects.
- *A set of cases:* concrete past problem solving experiences.

The approach for defining a taxonomical hierarchy depends strongly on the personal view of the domain. If a developer has a systematic top-down view of the domain, then it may be easier to use a top-down approach. A combined top-down and bottom-up approach is often the most feasible for many ontology developers, since the concepts “in the middle” tend to be the more descriptive concepts in the domain. A taxonomical and causal model for the drilling domain has been developed, in which all the entities are linked by binary relations. The causal model links the structured nodes together in a semantic network. The three main types of relation are: structural relations, e.g. has subclass; implication relations, e.g. causes; and associative relations, e.g. occurs in. To express the degree of coverage of the implication relations, quantifiers may be added to a relation, i.e. always, typically, sometimes, and occasionally. The “typically” quantifier is the default is none are used. Such a model facilitates model-based reasoning. In addition to the general knowledge, another important experience source in the oil well drilling domain is past cases.

In this study a case represents a hole cleaning problem. To make a case, all the relevant data and information are analyzed and a problematic situation is captured as a case. The successful response and repair activities are stored for the future use. Few cases were made in this study due to limited access to the data. The case-building process required quite a lot of data to fill a case’s features.

A case’s features consist of administrative data, wellbore formation characteristics, plan data, static and variable drilling data, the drilling activity performed before case occurrence, and response action and conclusion that provides a solution to the problem. The case structure is illustrated in Fig. 2.

Administrative Data	Wellbore Formation Characteristic	Activity Before Case Occurrence
Operator Company Well Identification Oil Field Identifier Drilling Contractor,...	Lithology Sandstone Siltstone ...	Case Series Start Time Possible Case Start Time .
<b>Drilling Operational Data</b>	Geological Period	<b>Response Activity Description</b>
Well Geometry Parameter Target Depth Section TVD ...	<b>Variable Data</b>	Case End Time Response Action...
Fluid Operational Parameter Mud Weight PV YP Water Activity Of Mud ....	Interpreted Activity Inferred Parameter Exposure time Openhole length ...	<b>Conclusion</b>
Drill String Parameter BHA Length Bit Run Number ...	Indicator Interpreted Event Packoff Tight spot ...	Final Section Consequence  Lesson Learned General Lesson Specific Lesson ....

Fig. 2. Case structure

### 3.2 Case Matching

The CBR cycle consists of four steps; retrieve, reuse, revise and retain. The retrieval task starts with a (partial) problem description, and ends when a best matching previous case has been found [12]. A similarity assessment process has been defined that can be run with or without the use of the causal model. The similarity method is an adaptation and extension of the Creek method [13, 14]. Our method consists of two



different similarity properties, one being direct or linear indexing, the other being concept abstraction. The latter is used when the model based module is utilized.

Basic similarity is computed by the following equation.

$$sim(C_{IN}, C_{RE}) = \frac{\sum_{i=1}^n \sum_{j=1}^m sim(f_i, f_j) \times relevance\ factor\ f_j}{\sum_{j=1}^m relevance\ factor\ f_j} \tag{1}$$

$C_{IN}$  and  $C_{RE}$  are the input and retrieved cases,  $n$  is the number of findings in  $C_{IN}$ ,  $m$  is the number of findings in  $C_{RE}$ ,  $f_i$  is the  $i^{th}$  finding in  $C_{IN}$ ,  $f_j$  the  $j^{th}$  finding in  $C_{RE}$ , and  $sim(f_1, f_2)$  is simply given as:

For symbolic concepts:

$$sim(f_1, f_2) = \begin{cases} 1 & \text{if } f_1 = f_2 \\ 0 & \text{if } f_1 \neq f_2 \end{cases} \tag{2}$$

$$sim(f_1, f_2) = 1 - \left| \frac{f_1 - f_2}{Max - Min} \right| \tag{3}$$

For linear concepts:

The relevance factor is a number that represents the weight of a feature for a stored case. The relevance factor of each feature was defined according to the feature’s importance for the case description. They were decided based on a survey among five experts to reduce subjectiveness of these values. The linear approach explicitly computes the values of similarity according to the minimum and maximum values of each concept. For example, minimum and maximum for true vertical depth has been set to zero and 8000 meter respectively. TVD for case 1 and 6 are 2869 and 2242 meter respectively, which provide 92 % similarity. Some of the indexing attributes will have both a symbolic and a linear description. An example of the categorization of numerical values is shown in Table 1. If a numerical value is available, linear similarity will be used and the symbolic terms will only be used in the model-based part.

**Table 1.** True Vertical Depth abstracted to symbolic entities

<i>True Vertical Depth (TVD)</i>	<i>Very shallow Well</i>	<1000 meter
	<i>Shallow Well</i>	1000-2000 meter
	<i>Medium Deep Well</i>	2000-3000 meter
	<i>Deep Well</i>	3000-4000 meter
	<i>Very Deep Well</i>	>4000 meter

### 3.3 Root Causes Assessment

The main objective is to determine the root cause starting out from three types of features: *Direct observations* – i.e. measurements, *inferred parameters* – i.e. values derived from observations, and interpreted *events* – i.e. particular concepts describing important states which require particular awareness or action. The features and causes are related through intermediate state concepts, see Fig. 3.

The model used is a semantic net-based model of entities linked by relations. Each relation is labeled. The root causes and the case features are all represented as entities in this model, and the model-based reasoner works by finding paths from the entities representing case findings to the entities representing root causes. Fig. 8 shows an example of two such paths (exampled in section 5).

The goal of the model-based reasoner is to determine which root causes or intermediate states are manifested by the features. Only some paths provide support for such a conclusion. In order to determine legal paths, plausible inheritance was used. This method is a generalization of normal subclass inheritance that allows inheritance of relationships over other relation types than ‘subclass of’ relations. Plausible inheritance is governed by a set of rules declaring which relation-types can be inherited over which relation-types. In this paper, causal relationships are transitive, and any relationship can be inherited over ‘subclass of’ relationships. For more information, see [13].

Assume there is a legal path from an observation to a root cause (see Fig. 3). Its strength is the product of the strength of each relation leading from the finding to the target entity [14].

$$Path\ strength = \prod_{i=1}^n relation\ strength_i \tag{4}$$

where n is the number of serial relations. Sometimes there is more than one explanatory path from different finding to each target entity (the root cause entity). The total explanation strength for each target entity is determined with Eq. (5). This calculated explanation strength will be a good indicator of being the possible root cause.

$$Explanation\ strength = 1 - \prod_{i=1}^m (1 - path\ strength_i \times weight) \tag{5}$$

where m is the number of paths. The strength of the indicating entities was decided based on a survey among five experts to reduce subjectiveness of these values. Weight of each indicating group, i.e. observation parameter, inferred parameters and events are 1/4, 1/2, and 1 respectively.

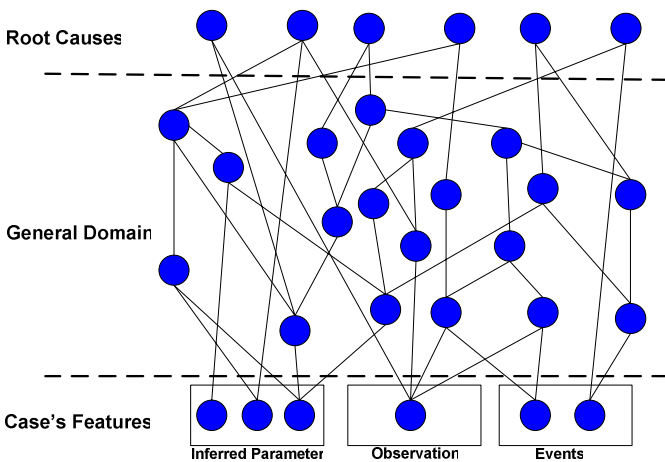


Fig. 3. Schematic model of the causal knowledge

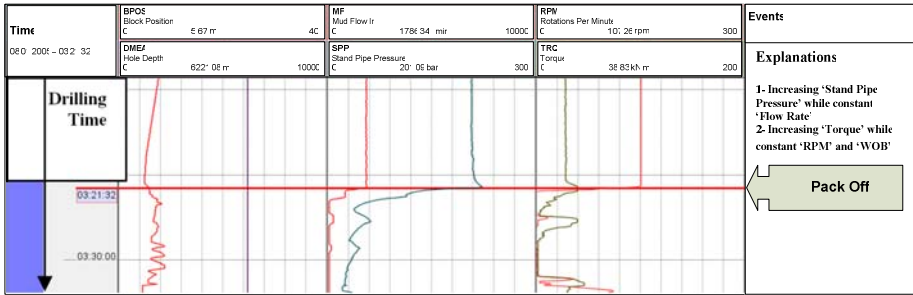


Fig. 4. ‘Pack Off’ recognition from observed data

Fig. 3 shows three different clusters, namely *events* e.g. ‘Pack Off’; *inferred parameters* e.g. ‘Open Hole Exposure Time’ (OHET); and *observation* e.g. ‘True Vertical Depth’ and ‘Mud Weight’ (density of the drilling fluid).

Observation factors include well plan data (drilling fluid and drill string parameters, well geometry), formation characteristic and case occurrence description.

In this section ‘Pack Off’ (of the event cluster) and ‘Open Hole Exposure Time’ (of the inferred parameter cluster) are exemplified. Fig. 4 shows a snapshot from one of the system’s screens, where ‘Pack Off’ event is interpreted from real time data. Observed data collected from sensors, like flow rate and stand pipe pressure, cannot explain the situation alone. They are more useful for case classification and for finding the root cause when combined. In the Explanations (right part of the figure), ‘Flow rate’ is the pump rate of drilling fluid for transportation of produced material from the bottom of the hole to the surface. ‘Stand Pipe Pressure’ is the pressure measured at the surface which may increase due to any obstacle inside the hole. Increasing of the ‘Stand Pipe Pressure’ will indicate a ‘Pack Off’ situation while the variables such as ‘Flow Rate’ are constant.

‘Open Hole Exposure Time’ (OHET) is one of the inferred parameters in this study, shown in Fig. 5. OHET is the time period when the formation is in contact with drilling fluid, which again may cause a problematic situation during the drilling operation. Higher exposure time can contribute to higher problems.

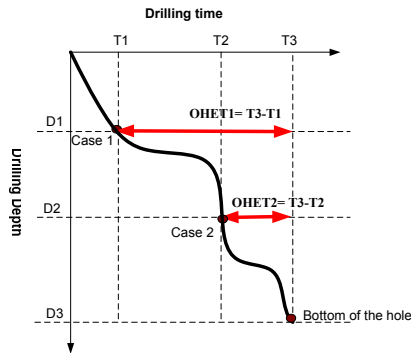


Fig. 5. Computation of the ‘Open Hole Exposure Time’ (OHET) for case 1 and case 2 when bit has reached D3

Selected points, i.e. the points where the cases were tagged by drilling time and drilling depth are shown in Fig. 5.

### 4 Case Matching Results

The case base contains seven cases related to poor hole cleaning problems experienced in North Sea wells. As mentioned, a symbolic and linear similarity framework was utilized. The case matching results for the case based module alone (CBR), the model-based approach alone (Model-Based) and for the integrated model- and case-based reasoning (KiCBR) will be presented. To evaluate the methods, a standard cross-validation technique is used, taking one case at a time out of the case base and matching against the six remaining cases. Fig. 6 shows the case matching results for case 7 and case 5 as unsolved cases. For case 7, the retrieved case with highest similarity was case 2 with 18% similarity using the CBR method. When the KiCBR method was applied instead, case 3 was retrieved with 39% similarity.

In order to differentiate between the retrieved cases, they were grouped into three levels according to severity (how much drilling downtime they caused). The three levels of downtime are; insignificant, significant and highly significant repair time.

For instance, evaluation of downtime for case 7 revealed that this case required highly significant repair time while cleaning the hole. However, the CBR method retrieved case 2, which had insignificant repair time. On the other hand, the KiCBR method retrieved case 3 which is more similar to case 7.

In another example, the case matching process was run for case 5 as an unsolved case. When using the CBR matching method, case 2 was retrieved, while case 6 was retrieved using KiCBR. Case 2 and case 6 are grouped in the same class in terms of the downtime during the drilling, but detail study showed that case 6 had significant downtime later in the operation around the same area, and this is similar to the situation in case 5. This means that case 6 is more similar than case 2.

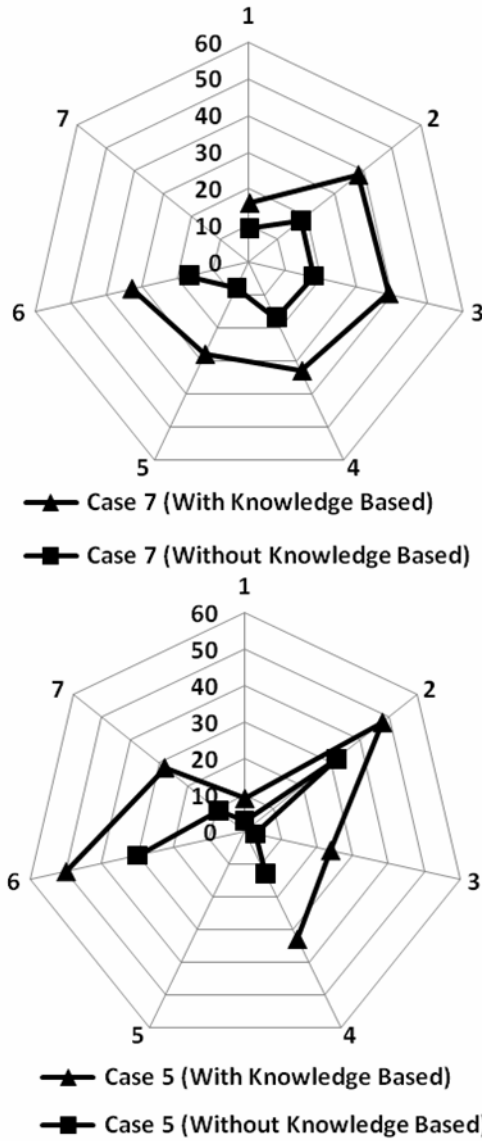
The results show not only improvement for similarity assessment but also good prediction in problem solving. The effect of including general knowledge was monitored by changing not only the similarity but also the retrieved cases.

Similarity assessments are summarized in Fig. 7. The similarity growth was fluctuating from about 20 % to 100 % or even higher. The most similar case by means of case-based, model-based and KiCBR are summarized in Table 2.

**Table 2.** Similarity assessment by means of different reasoning methods

Unsolved case	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
Retrieved by Case-based	Case 3	<b>Case 6</b>	Case 4	<b>Case 3</b>	Case 2	<b>Case 2</b>	Case 2
Retrieved by Model-based	<b>Case 7</b>	Case 7	<b>Case 7</b>	Case 5	Case 4	Case 4	<b>Case 3</b>
Retrieved by KiCBR	Case 3	<b>Case 6</b>	<b>Case 7</b>	Case 6	<b>Case 6</b>	<b>Case 2</b>	<b>Case 3</b>

**Bold** items in the above table represent the best case for each unsolved case according to downtime and detail studies. KiCBR was able to retrieve the optimal case in 5 out of 7 cases, while model-based and case-based retrieved only 3 optimal cases.



**Fig. 6.** Case matching results (in %) for the case based module alone, and for the combined case based and model based module. Unknown case 7 (upper) and case 5 (lower) were matched against the remaining 6 cases. Lines between points are only used for better illustration.

In summary, two important phenomena can be observed from the above tests.

First, the general knowledge can generally increase the similarity for all cases in different rates.

Second, general knowledge may also change which case obtains the highest similarity.

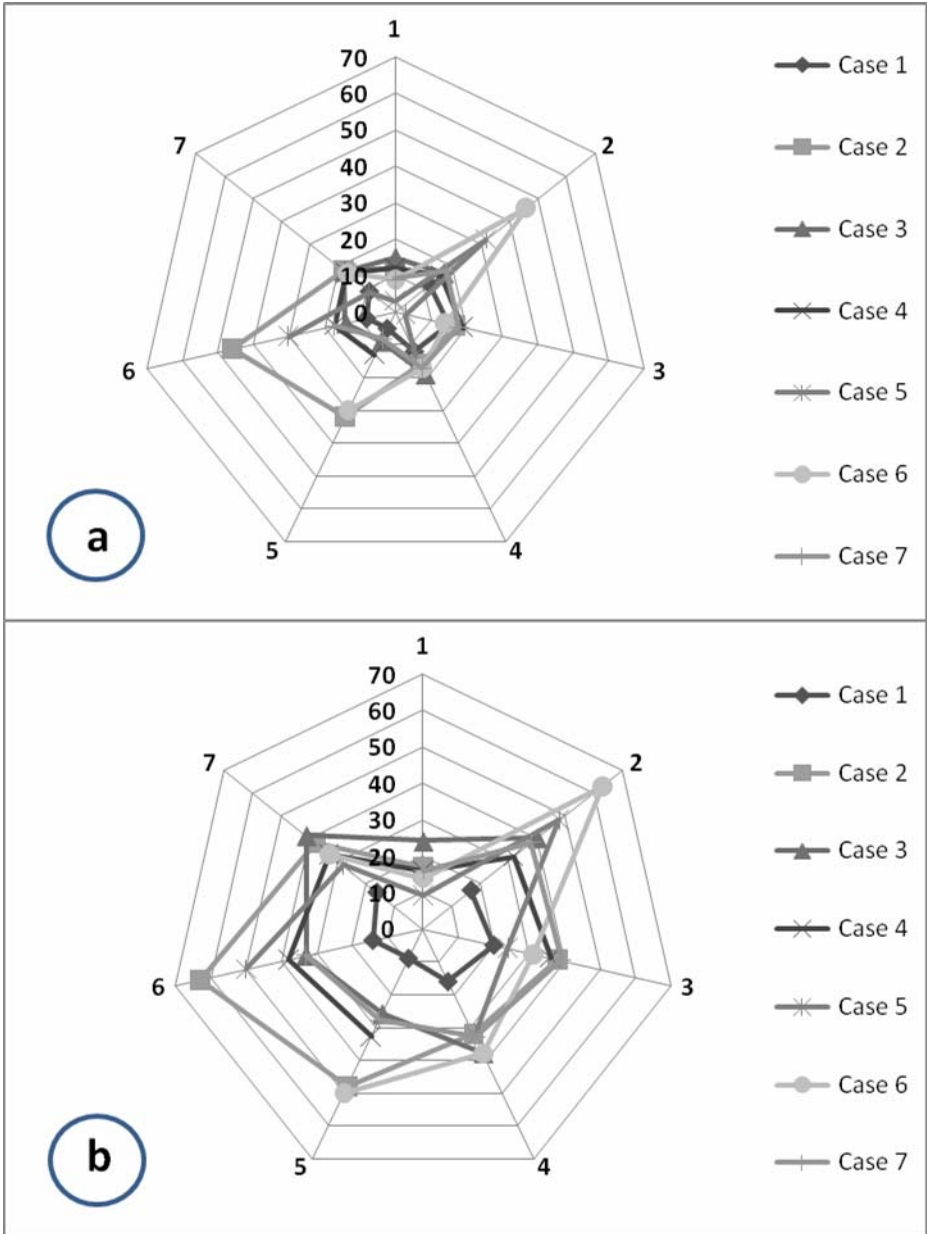
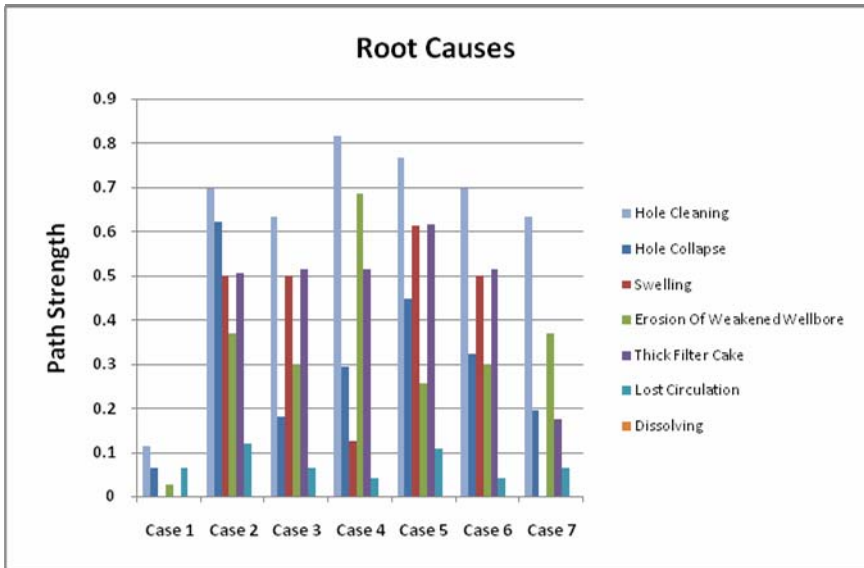


Fig. 7. Case matching using CBR without knowledge model (a) and with knowledge model (b)





**Fig. 9.** Path strength of 7 cases based on general model to determine level of the hole cleaning problem

*Second plausible inheritance path:*

‘Open Hole Exposure Time’ has subclass ‘Long Exposure Time’ causes sometimes ‘Erosion Of Weakened Wellbore’.

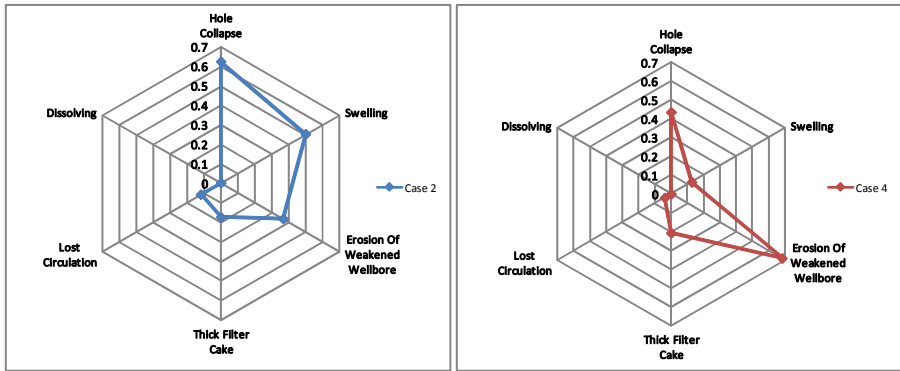
This link starts out from ‘Open Hole Exposure Time’ which is an inferred parameter. The path strength for this explanatory path is 0.5.

For each root cause, all plausible inheritance paths from each inferred or observed parameter in the cases is combined using Eq. (5), which determines the explanation strength. This calculation yields a number between 0 and 1 for each root cause, with a higher value indicating higher support for that root cause. Fig. 9 presents the value for each root cause for each of the seven cases.

Textual sources written during or after the drilling operation (Daily Drilling Report (DDR) and End of well report (EWR)) as well as real-time sensor logs showed us that six of the seven cases were highly representative of hole cleaning problems. As shown in Fig. 9 derived path strength of all seven cases points at poor hole cleaning except for case 1. In case 1, no events and inferred parameters took place. Therefore, explanation strength is based on just observed parameters, which results in a fairly low value of the path strength.

Once the root cause is found, it can be treated by applying a repair action. Each problem needs to be treated differently. A preliminary assessment of well data was performed to determine the specific root cause. In figure 10, the results for two cases (case 2 and 4) are shown. The plausible inheritance model provides strongest support for the ‘Hole Collapse’ and ‘Erosion Of Weakened Wellbore’ to be the specific root causes of poor hole cleaning for case number 2 and 4 respectively. Dissolving is zero for all the cases since there was not any salty rock in the studied holes. The presence





**Fig. 10.** Finding of root causes by means of knowledge model for case 2 (left) and case 4 (right)

of claystone (i.e. a type of rock) and about 26 days of ‘Open Hole Exposure Time’ caused the claystone to react with drilling fluid and the formation around the hole wall was eroded.

One of the main purposes of introducing knowledge based system is to advise the user of how to modify the controllable drilling parameters with respect to the associated root cause. Whenever the cause of a problem is revealed, the proper remedy can be applied. ‘Hole Collapse’ is one of the major causes of poor hole cleaning, mostly resolved by adjusting the density of the drilling fluid (mud).

Indications so far although based on a small case base, indicate that the KiCBR method may be better at retrieving the correct case, but even where this method is used, the explanation facilities of the model-based approach is valuable, as it allows the user to see what factors contribute to the problem by providing explanations. The model-based approach also calculates the support for different root causes independently, allowing it to conclude that multiple problems can be present. This is important as multiple problems requires multiple or complex remedies. For instance, for case 2 in Fig. 10 ‘Swelling’ has high support, although ‘Hole collapse’ has even higher support. Chances are, both of these problems are present.

## 6 Conclusion

The application of a relatively new methodology to reduce downtime during the oil well drilling has been considered. A combination of symbolic and linear similarity was utilized. Case similarity was changed by combining case based and model based reasoning.

KiCBR obtained a higher similarity and accuracy than case based reasoning alone. Similarity between an unsolved case and cases in the case base increased in average by typically 50 % after introducing the knowledge module in the reasoning process.

The most probable root cause could be determined on basis of the knowledge model. The root cause determined with the model-based approach had a good correlation with the expert analysis from real-time sensor data.

## 6.1 Further Work

The results indicate that combining a causal model with case based reasoning improves case matching. Furthermore, the knowledge model serves as explanatory support for finding root causes. But in this study few cases were available and the results have to be tested out with more cases as soon as more data will be available. Our aim is to implement this platform on more cases and perform a broader and more detailed assessment of the methodology.

## Acknowledgments

The work reported here has been partly funded by StatoilHydro and the Research Council of Norway under the Petromaks program, contract no. 169463/S300. The authors would like to express appreciation to people from StatoilHydro, through its project leader Erik Nyrnes, and colleagues from Verdande Technology, for their contributions to the results presented.

## References

1. Datta, B.K., Ratnayake, C., Saasen, A., Omland, T.H.: Hole Cleaning and Pressure-Loss Prediction From a Bulk Transport Perspective. Paper SPE 96315 presented at Offshore Europe, 6-9, September 2005, Aberdeen, United Kingdom (2005)
2. Skalle, P., Sveen, J., Aamodt, A.: Improved Efficiency of Oil Well Drilling through Case-Based Reasoning. In: Mizoguchi, R., Slaney, J.K. (eds.) PRICAI 2000. LNCS, vol. 1886. Springer, Heidelberg (2000)
3. Mendes, J.R.P., Morooka, C.K., Guilherme, I.R.: Case-based reasoning in offshore well design. *Journal of Petroleum Science and Engineering* 40, 47–60 (2003)
4. Popa, A., Popa, C., Malamma, M., Hicks, J.: Case-based reasoning approach for well failure diagnostics and planning. SPE paper 114 229, presented at the 2008 SPE Western Regional and Pacific Section AAPG Joint Mtg, Bakersfield, March 31- April 2 (2008)
5. Abdollahi, J., Carlsen, I.M., Randhol, P., Tenold, E., Haga, H.B., Jakobsen, T.: A Case-Based Approach to Understand the Complexity of Causal Connections Related to Well Integrity Problems. In: IADC/SPE 111129, Presented at the 2008 IADC/SPE Drilling Conference held in Orlando, Florida, U.S.A., March 4–6 (2008)
6. Yu, M., Takach, N.E., Nakamura, D.R., Shariff, M.M.: An experimental study of hole cleaning under simulated downhole conditions. In: Paper SPE 109840 presented at SPE Annual Technical Conference and Exhibition, Anaheim, California, November 11-14 (2007)
7. Lapierre, S., Courville, G., Song, J.: Achieving Technical Limits: Expanded Application of Real-Time Pressure-While-Drilling Data Helps Optimize ROP and Hole Cleaning in Large-Diameter, Directional Intervals. In: Paper SPE 99142-MS presented at IADC/SPE Drilling Conference, Miami, February 21-23 (2006)
8. Adari, R.B., Miska, S., Kuru, E., Bern, P.: Selecting Drilling Fluid Properties and Flow Rates For Effective Hole Cleaning in High-Angle and Horizontal Wells. In: Paper SPE 63050-MS presented at SPE Annual Technical Conference and Exhibition, Dallas, October 1-4 (2000)

9. Sanchez, R.A., Azar, J.J., Bassal, A.A., Martins, A.L.: Effect of Drillpipe Rotation on Hole Cleaning During Directional-Well Drilling. *SPE Journal* 4(2), 101–108 (1999)
10. Charlez, P.A.: *Rock Mechanics: Petroleum Applications*, p. 661. Editions Technip. (1997)
11. Peden, J.M., Ford, J.T., Oyenyin, M.B.: Comprehensive Experimental Investigation of Drilled Cuttings Transport in Inclined Wells Including the Effects of Rotation and Eccentricity. In: Paper SPE 20925-MS presented at European Petroleum Conference, The Hague, October 21-24 (1990)
12. Aamodt, A., Plaza, E.: Case-Based Reasoning: Fundamental Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications* 7(1), 39–59 (1994)
13. Sørmo, F., Aamodt, A.: Knowledge elaboration for improved CBR. In: Sixteenth International Joint Conference on Artificial Intelligence, Workshop ML-5: Automating the Construction of Case-Based Reasoners, Stockholm, pp. 39–43 (1999)
14. Aamodt, A.: Knowledge-Intensive Case-Based Reasoning in CREEK. In: Funk, P., González Calero, P.A. (eds.) *ECCBR 2004. LNCS*, vol. 3155, pp. 1–15. Springer, Heidelberg (2004)

# Author Index

- Aamodt, Agnar 450, 509  
Adeyanju, Ibrahim 14  
Aha, David W. 29, 434  
Aleven, Vincent 45  
Alonso, Rafael 465  
Althoff, Klaus-Dieter 389  
Ashley, Kevin 45
- Bach, Kerstin 389  
Badra, Fadi 60  
Bianchi, Reinaldo A.C. 75  
Bierer, Annett 90  
Blanzieri, Enrico 328  
Bottrighi, Alessio 225  
Bramsen, Diane 465  
Bridge, Derek 120, 479  
Briggs, Peter 494
- Chakraborti, Sutanu 270  
Cojan, Julien 105  
Cordier, Amélie 60  
Coyle, Maurice 494  
Craw, Susan 1  
Cummins, Lisa 120  
Cunningham, Pádraig 328
- Delany, Sarah Jane 135  
Derbinsky, Nate 403  
Díaz-Agudo, Belén 313, 418
- Escalera, Sergio 298  
Esfandiari, Babak 150
- Floyd, Michael W. 150  
Fornells, Albert 418  
Fornells, Eduard 418  
Funk, Peter 358
- Gay, Pablo 195  
Golobardes, Elisabet 418  
González-Calero, Pedro A. 313  
Gupta, Kalyan Moy 434
- Hofmann, Marcus 90  
Hogg, Chad 465
- Kendall-Morwick, Joseph 165  
Khemani, Deepak 270  
Kofod-Petersen, Anders 450
- Laird, John E. 403  
Lamontagne, Luc 14  
Leake, David 165  
Lee-Urban, Stephen 180  
Leonardi, Giorgio 225  
Li, Hua 465  
Li, Lei 374  
Lieber, Jean 60, 105  
López, Beatriz 195  
Lopez de Mantaras, Ramon 75  
Lothian, Robert 14  
Lynch, Collin 45
- Mateas, Michael 343  
Mehta, Manish 210  
Molineaux, Matthew 29  
Montani, Stefania 225  
Moore, Philip 434  
Muñoz-Avila, Héctor 180, 465  
Murphy, Glen 479
- Nugent, Conor 479
- O'Mahony, Michael P. 494  
Ontañón, Santiago 210, 240  
Øyen, Bernt-Håvard 479
- Pantazi, Stefan 256  
Pinkwart, Niels 45  
Pla, Albert 195  
Plaza, Enric 240  
Portinale, Luigi 225  
Pous, Carles 195
- Radeva, Petia 298  
Raghunandan, M.A. 270  
Ram, Ashwin 210  
Recio-García, Juan Antonio 418  
Reichle, Meike 389  
Rissland, Edwina L. 6  
Rohrer, Brandon 285  
Ros, Raquel 75

- Salamó, Maria 298  
Sánchez-Ruiz, Antonio A. 313  
Segata, Nicola 328  
Skalle, Pål 509  
Smyth, Barry 494  
Sørmo, Frode 509  
Sripada, Somayajulu 14  
Sukthankar, Gita 29
- Terenziani, Paolo 225
- Valipour Shokouhi, Samad 509
- Weber, Ben G. 343  
Wiratunga, Nirmalie 14
- Xiong, Ning 358
- Yang, Qiang 374
- Zhuo, Hankui 374